## DOCTOR OF ENGINEERING (ENGD)

# Investigating how computational tools can improve the production process of stop-motion animation

Howell, Lindsey

*Award date:*
2015

*Awarding institution:*
University of Bath

# Investigating how Computational Tools can Improve the Production Process of Stop-Motion Animation

submitted by

Lindsey Elaine Howell

for the degree of Doctor of Engineering

of the

University of Bath

Department of Computer Science

August 2014

# Contents

# List of Figures

# Acknowledgements

A huge thank you to my academic supervisor, Peter Hall and my industrial supervisor Philip Child, for their guidance and support. Thank you both for sharing your knowledge and time so generously. I would also like to thank Jess McKillop and Darren Cosker for their help and support throughout the process.

I am grateful to the Centre for Digital Entertainment at the University of Bath, Aardman Animations Ltd. and the EPSRC for supporting this research. With thanks to the Centre for Digital Entertainment, not only for funding my research, but also for the ongoing support of the CDE team and for the opportunities they have provided.

I am very grateful to Aardman Animations for the opportunity to work in their company and for allowing me access to the wonderful work they do. A particular thank you to those people in the company who helped me with a variety of unusual requests, in order to get my research projects up and running.

With very special thanks to my amazing family and amazing friends, who have supported me through the highs and lows and without whom I would never have reached this point. To my key support team, you are the best.

# Summary

Stop-motion animation is a traditional form of animation that has been practised for over 100 years. While the unique look and feel of stop-motion animation has been retained in modern productions, the production process has been modernised to take advantage of technological advancements. Modern stop-frame animation production integrates digital imaging technology and computational methods with traditional hand-crafted skills.

This portfolio documents three projects undertaken at Aardman Animations, each investigated with the aim of improving efficiency in the stop-motion production process:

- **Rig removal** is the removal of equipment, or 'rigging', used on set during stop-motion animation to hold characters or objects in unstable positions. All rigging captured in frames must be removed in post-production and currently manual methods are used which can be very time-consuming. The key task is to separate the character from the rig. In Chapter 2, I present a novel spatio-temporal segmentation algorithm for segmenting characters from stop-motion footage. The algorithm has been designed to work with stop-motion animated content, in contrast to other state of the art algorithms which struggled when tested on stop-motion footage.

- **Set shift** is a problem which occurs when background items on set move subtly over the time taken to shoot a scene. For example, temperature and humidity changes can cause wood to warp during a weekend, changing the position of a background object the following week. These small 'shifts' are recorded in the footage and must be corrected in post-production. Chapter 3 describes the problem in detail, investigates potential solutions and explains why solving set shift automatically is a significant challenge.

- **Plasticine shading** is required when a plasticine model has to be generated computationally. One motivation for producing footage computationally is that problems such as rig removal and set shift do not arise. In order to simulate plasticine accurately, the distinct reflectance model of this material must be known and reproduced. By collecting experimental data from plasticine samples and fitting parametric models, I have developed a bespoke surface shading model for plasticine (Chapter 4). This new model provides the best fit to the measured data when compared to existing state of the art surface shaders. It has been implemented into commercially used production systems, for use with existing rendering software.

Advancing state of the art research is only one of the challenges when working in a production studio such as Aardman Animations. Additionally, findings must be integrated into the production pipeline. Chapter 5 discusses the challenges and constraints faced when conducting research in this environment.

In order for stop-motion animation to remain competitive it is vital that production companies stay up-to-date with technological advancements in research areas that can contribute to their production processes. I conclude by discussing whether technological advancements can help Aardman Animations in improving the efficiency of their stop-motion production pipeline.

# Chapter 1

# Introduction and Background

## 1.1 Introduction

This research portfolio investigates how technical research can be used to make the production process of stop-motion animation more efficient. Stop-motion animation is a relatively traditional form of animation where much of the content is created manually using a series of photographed images. It involves capturing each frame of footage individually and collating the frames together in sequence to form animated content.

Aardman Animations commissioned a project with the University of Bath to conduct research into the development of computational tools, to see if improvements can be made in a number of different areas of the production process. They wanted to analyse the potential benefits of technical research and to understand whether there is any competitive advantage to using such technology. This document explores whether state of the art research in computer graphics and computer vision can be incorporated into the traditional world of stop-motion animation to improve the production process. Several different projects were investigated over the course of this research and the timescales for the key research projects are shown in table 1.1.

| **Project title** | **Start date** | **End date** |
|---|---|---|
| Rig removal (chapter 2) | September 2011 | March 2013 |
| Set shift (chapter 3) | March 2013 | June 2013 |
| Plasticine shading (chapter 4) | June 2013 | June 2014 |

Table 1.1: The timescales of each of the research project completed.

The first project investigates rig removal (chapter 2). Rigs are used in stop motion production to support characters and objects in unstable positions. For example, if a character is jumping then a rig is used to hold the character in mid-air while shots are captured. This rigging equipment must then be removed in post-production. The rig must be removed from every frame in which it appears and currently manual techniques are used. The most important area to separate is the join between the rig and the character to which it is attached. State of the art segmentation research is reviewed and

(a) Results from the geodesic segmentation algorithm of Criminisi et al. (2010).

(b) Results from the algorithm presented in this portfolio.

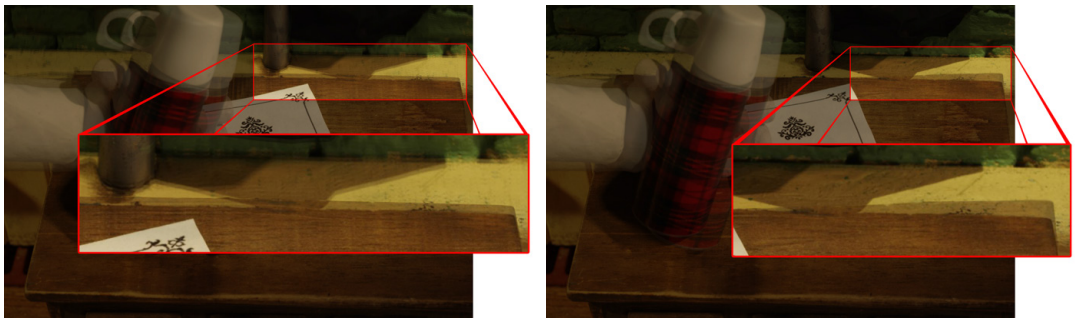
Figure 1-1: A comparison of results from one state of the art video segmentation algorithm and the new video segmentation algorithm presented in this portfolio, when tested on production footage ©Aardman Animations Ltd 2014.

tested to see how well it can extract the character from the rig. This project investigates techniques to segment an object from stop-motion video footage and presents a new contribution to the literature with an improved segmentation algorithm for video content. The results show a significant improvement in accuracy when compared to a state of the art segmentation algorithm (figure 1-1).

The second project investigates potential solutions to the problems associated with set shift (chapter 3). Set shift occurs in stop-motion animation when objects in the scene shift or warp over the course of recording the animated content. For example, if a set is left for a weekend part way through a shot, wooden planks can warp by the time the next frames are captured. These shifts are subtle and are not visible when recording the frames but become obvious when the content is played back in sequence. The shifts have to be corrected in post-production, with accidental movements in the scenery removed. This task can be complex and therefore time-consuming. A variety of methods of shift detection were investigated and a prototype application was built to test potential software solutions to the problem of set shift. The methods tested, and challenges associated with the complexity of the set shift problem, are reviewed in section 3.4.

The third research project investigates the material properties of plasticine. Plasticine is a key material that is often used by the company for their 'claymation' style of stop-motion animation. In addition to using plasticine on set, it is common for plasticine to be simulated computationally for computer generated models. A standard shading model has previously been used to represent the surface properties of plasticine. The research presented in chapter 4 investigates whether this standard model is a true representation of plasticine materials. To analyse the distinct surface properties of plasticine materials, experimental data was collected from plasticine samples to populate a BRDF (Bi-directional Reflectance Distribution Function) table (section 4.3.1). Existing analytical shading models were tested for compatibility to the measured data using mathematical fitting. It was found that plasticine has some distinct physical properties in the way it reacts to light and a bespoke shading model was developed to simulate these specific surface properties. The research shows that the novel plasticine model provides the best fit to the captured data and simulates the behaviour more closely than existing state of the art models. The results of the new simulated plasticine model are shown in figure 1-3.

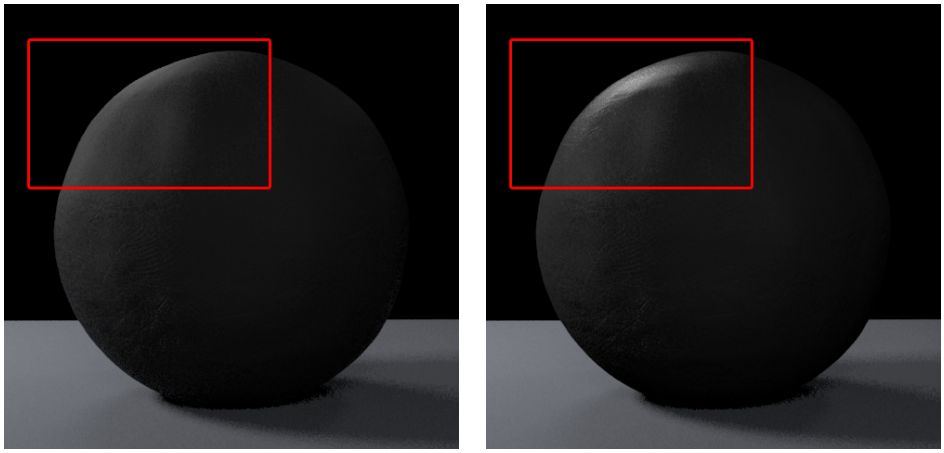
For each of the research areas discussed I present details of the research completed and the findings



(a) A scenario where the background has shifted slightly.

(b) A scenario where the edge of a wooden table has warped.

Figure 1-2: Set shift examples which have been exaggerated for demonstration purposes. The source image overlays the target image to make the differences visible. The set shift movements are highlighted and magnified. ©Aardman Animations Ltd 2014.



(a) Results using the out of the box surface shader used in production.

(b) Results using the new PlasticinePlus surface shader developed on the physical properties of plasticine.

Figure 1-3: A comparison of results from the benchmark production surface shader and the new PlasticinePlus surface shader developed in this portfolio.

from the investigations. All research was conducted in industry, meaning that extra limitations were imposed due to industrial constraints. For example, the tools created needed to integrate with the current production pipeline and they needed to be useable by production artists. Chapter 5 discusses the additional considerations and work that had to be undertaken in order to tailor the research contributions to meet the requirements of an effective industrial process. It also discusses two smaller production projects that were undertaken in conjunction with the larger research projects.

Researching three different areas of stop-motion animation for Aardman Animations has produced a number of significant findings. Improvements and contributions have been made to the literature in the areas of video segmentation and in surface shading. Additionally research into set shift has answered

the question set by the company, about whether the development of a tool in this area would be worth investing in at this stage. These contributions help the company in their understanding of the research topics and of the tools available. As new research is published they will know the quality of result that they require in order for it to be useable in their production process. If the research is at the level required then it can be implemented and integrated into the production pipeline using the structure outlined by the projects completed in this portfolio.

I conclude by addressing my original research question as to how technical tools can help to improve the production of traditional stop-motion animation and summarising the contributions of each research project. The success of this research is analysed along with the challenges and complexities of combining a traditional animation form with state of the art computational research (chapter 6). Overall, this research demonstrates that technical tools can help with improving the production of stop-motion animated content, but it is important to choose the areas of research carefully. There are research areas which can be improved with technical tools, but they are specific and results must be to the standard required by industry. There are other research areas which are not currently advanced enough for use in production; this was exemplified in the rig removal project. However, it is important that Aardman Animations keep current with these research areas, so that when content of the correct quality is released, it can be used to streamline the production process.

### 1.1.1 Publications

L. Howell, P. Child, P. Hall (2013) *"Improved Image Segmentation Using Motion"*
In: CVMP '13 Proceedings of the 10th European Conference on Visual Media Production,
Article No. 1. ACM New York, NY, USA.
ISBN: 978-1-4503-2589-9 DOI:10.1145/2534008.2534012

L. Howell, P. Child, P. Hall (2014) *"Plasticine Shading"*
In: CVMP '14 Proceedings of the 11th European Conference on Visual Media Production,
Article No. 4. ACM New York, NY, USA.
ISBN: 978-1-4503-3185-2 doi¿10.1145/2668904.2668933

## 1.2 Background

### 1.2.1 Animation Production and the Production Pipeline

Animated video content can be generated in a number of ways. Some example methods are hand-drawn 2D cell animation, 2D or 3D computer animation or stop-motion animation. Each method, and each studio, has its own production pipeline but there are key stages in the process.

- **Pre-production: designing the animation**
  The first stage is to design what will happen in the final animated content and how it will look.

Typically this involves activities such as storyboarding, character design, set design and script-writing. Once the final design has been approved, artists can begin work on the production of the content.

- **Production: generating the content**
  For stop-motion production, characters and sets need to be built and lighting and cameras must be set up. The characters and props are animated while capturing the scene frame by frame. For computer generated animation characters, props and sets need to be modelled and textured. Moving objects within the shot need to be rigged and animated. The scene needs to be lit and the final frames are produced by rendering the animated content from virtual cameras.

- **Post-production: adding to the content**
  Post-production work is completed on the captured or rendered frames from both the stop motion and computer generated imagery. Examples of the type of work completed in post production are the addition of visual effects, removal of unwanted objects in the scene and compositing of two or more images together.

- **Transcoding: to produce the final output**
  Once all work has been completed on the footage, the frames are output to a video in the chosen format and combined with the audio content.

Aardman Animations produce content using a number of the different methods mentioned. Figure 1-4 gives an overview of the production pipeline and where the completed research projects fit within the structure. The research in this portfolio was completed for the CGI (Computer Generated Imagery) department, which covers the production of computer generated animation and post-production for both computer-generated and stop-motion animation. In this research I focus on challenges associated with stop-motion animation. As demonstrated by the diagram, the processes used during the production of stop-motion footage are protected and so could not be changed. It was vital that research did not interfere with creativity and that the methods used on set remained traditional.

### 1.2.2 Stop Motion Production

Aardman Animations are famous for their stop-motion animated productions and in particular for using plasticine to create and manipulate characters. Characters such as Wallace and Gromit and Morph are much loved and well-known all over the world. Stop-motion, or stop-frame, animation is a traditional form of animation with examples of the technique used in the "The Humpty Dumpty Circus" as early as 1897 (Guinness World Records, 2014). Aardman Animations began producing animated content using stop-motion techniques in 1972, with examples of their early work being "Morph" and "Conversation pieces".

Stop-motion animation is a method of animation in which the content is captured by recording individual static frames of footage which, once played in sequence, give the illusion of moving imagery. Stop-motion production is traditionally a method of animation which requires very little technical input. However, as techniques have become more sophisticated, computational methods are used increasingly in many areas of the production pipeline.

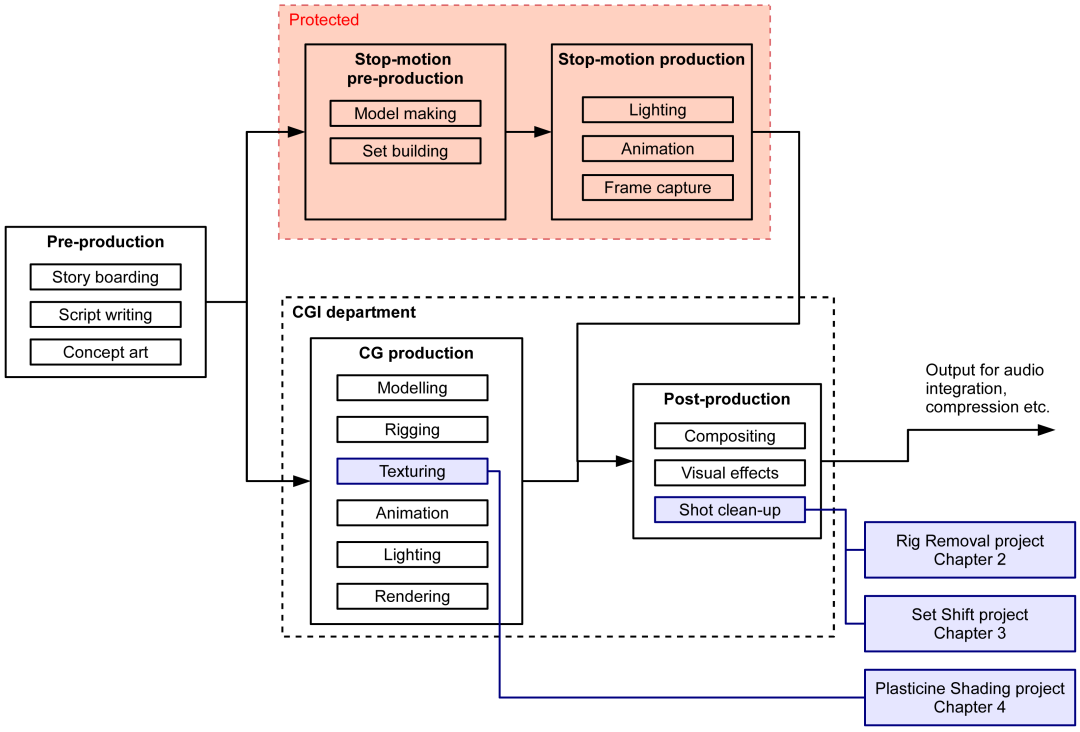



Figure 1-4: Production overview showing the production pipeline at the company. The diagram shows the role of the CGI department, where the final research projects fit within the production process and the protected areas of stop-motion animation with which research work could not interfere.

The techniques used for the animation of stop-motion footage remain similar to the traditional methods used. Each frame is captured in the studio while small movements are made to the characters and objects in the scene. However, in other areas technological advances have made changes to the methods, for reasons such as increased flexibility or efficiency. One major addition of digital techniques to stop-motion animation was the introduction of digital cameras, replacing traditional analogue cameras. This gives much more flexibility and immediate feedback for those working on set. Pete Lord, co-founder of Aardman Animations comments on the transition "Before digital cameras, an animator would have to wait for the lab to develop the film to see if the shot turned out the way they hoped, and if it didn't, they'd sometimes have to redo an entire scene which could take weeks. With digital cameras (like the Canon 5D) now being used to shoot the movie, an animator can see the results immediately, so if a frame is off or a problem arises they can fix it right away." (Weintraub, 2012). This demonstrates one example of a traditional method of animation embracing technological advancements to improve the process. This portfolio investigates whether technological research can be incorporated into other areas of stop-motion animation to improve the production process.

As mentioned previously, the processes associated with stop-motion capture can not be affected or changed by any research implemented. However, there are tasks in the production pipeline in which computational tools can be implemented. One area in which research can be implemented, is the

post-production stage. After capturing imagery on set, all frames from a stop-motion production are processed in post-production using compositing software. Once post-production work has finished, the individual frames are processed by the 'Edit' department, from which footage is output as the final combined audio and video content. The stages of post-production and 'Edit' have been significantly impacted by advancements of technology. Therefore, although the way of creating the footage remains traditional, some of the processes surrounding it have become more computationally-driven. In this research portfolio, the aim is to find areas of stop-motion production which can be improved using the latest computational knowledge and research.

### 1.2.3 Motivation for the Academic Research

Aardman Animations are interested in conducting research into specific areas of their production pipeline. The motivation for this portfolio of research is to discover whether computational methods can be used to improve production processes. In a wider context, they are interested in collaborating with academia for two reasons. Firstly, they would like to understand the potential influence of academia on the animation industry and the potential impact of collaborations. Secondly, they are interested in ways of evolving teaching curriculums so that they are more industrially-based.

There were several problem areas outlined by the CGI department as areas that they would like investigated. As Aardman Animations does not have its own dedicated research department or is of a size to develop blue sky research, they were keen to investigate these areas with the backing of an academic support network.

### 1.2.4 Research Projects

There were a number of research areas that were outlined for investigation:

- Rig removal: A post production problem of extracting the rigging equipment which is captured in frames. Each piece of rigging must be removed and replaced with the background content.
- Set shift: A post production problem of correcting objects that have shifted or warped over time on set. These shifts become obvious when the images are played in sequence and motion can be seen in areas where there should not be movement.
- Skinning: Improvement on the current automatic skinning techniques used in CG rigging, with the aim being to reduce the amount of manual input.
- Fluid simulation: Improving fluid simulation algorithms.
- Plasticine Shading: Creation of a surface shader to simulate the properties of plasticine and how it reacts to light.
- Depth extraction: Generation of depth information from frames captured on set. Understanding the depth of objects on set would provide more information about the scene in the post-production stages. This research could contribute to solutions for problems such as rig removal and set shift.

An additional constraint of the research was that projects must generate both novel academic research as well as functional tools for use in industry. As the company does not have an established research department it has been a challenge to find the projects which satisfy both requirements. Those projects listed above demonstrate some of the subject areas where it was felt there is an overlap between academic novelty and practical application in a stop-motion animation studio.

The decision was made to focus on stop-motion production and the projects investigated in detail in this portfolio are those of rig removal (chapter 2), set shift (chapter 3) and plasticine shading (chapter 4). The situation of these projects within the production pipeline is shown in figure 1-4. Having looked at a range of research projects to aid with the production of stop-motion animation, I discuss whether technical tools can help with the production of a traditional form of animation such as stop motion (chapter 6).

As this research was completed in an industrial environment, additional limitations needed to be considered. Any computational tools developed had to work independently of the way that stop-motion artists work on set. Developed tools had to be integrated into the existing production pipeline and with its associated software (chapter 5).

# Chapter 2

# Rig Removal

## 2.1 Introduction

When animating in stop-motion, rigging equipment is often used to get characters or objects into the correct positions. For example, if a character is in mid-air or is precariously balanced then some additional material will be used to hold the character in place. This material is known as "rigging". Rigging is often visible in frames captured in production and has to be manually removed in post-production. In order to remove the rigging an accurate segmentation method is needed to separate the character from the rig.

Currently, rigging objects are predominantly removed using rotoscoping. This involves the user creating a mask over the rig object for every frame. Rotoscoping is a time-consuming and labour-intensive part of the production process which can consume "up to 20% of the human time required for a CGI project" as cited in Agarwala et al. (2004b). For the task of rig removal the rigging object needs to be outlined on every frame in order to separate it from the character or object being held by the rig. The most important task is to get a clean separation between the object to which the rig is attached and the rigging itself. To do this one can either mask the rigging or the object. Masking the object helps to remove the rig because the background region, including the rigging, can be replaced with information from the clean plate. A clean plate is captured at production time and contains the background scene with no foreground objects. The motion of the rig is unpredictable and it can move randomly from frame to frame. In contrast, the motion of the object is coherent. I have proposed a new improved segmentation method which uses this motion information to extract the character or object to which the rig is attached.

Research into state of the art segmentation methods and current production tools shows that some form of user interaction will be necessary. The rigs and characters have no common appearance so there is no way of classifying and detecting objects in the scene. The chosen method of user interaction requires the user to indicate the object and background regions with simple user strokes. In order to make use of the coherent motion information, the object attached to the rigging will be segmented. User interaction is required for the item attached to the rig to be marked with "object area" user strokes and the rest of the image, including the rigging, to be marked with "background area" user strokes.

Figure 2-1: Results using the new algorithm for character segmentation on three frames of stop-motion footage. ©Aardman Animations Ltd 2014.

Figure 2-2: Results from a state of the art segmentation algorithm (Criminisi et al., 2010) using identical input to figure 2-1. ©Aardman Animations Ltd 2014.

A useful production tool will not only separate the object from its surroundings for a single frame, but also accurately follow the segmented object as it moves during stop-motion video. Stop-motion footage is inherently different in motion to that of live-action footage. The movements in stop-motion animation are discontinuous rather than the smoother nature of live-action video. For this problem one can assume that the object to be segmented will be in motion, because if the object has not moved then the mask from a previous frame can be used to extract the object. As the nature of the stop-motion footage at Aardman Animations is known, I can make two assumptions based on the typical content of frames. Firstly, that the images will contain no motion blur and secondly, that the camera will be static throughout the scene.

Existing segmentation algorithms struggle when tested on production footage. In particular, methods that rely on colour analysis can suffer when object and background colour ranges are similar. One state of the art algorithm by Criminisi et al. (2010) was looked at and it was found that the jerky nature of stop-motion footage and the method's reliance on colour made accurate segmentation difficult. Images with similar object and background colour ranges produced unsatisfactory segmentation results. To improve these aspects of the algorithm I have extended the method to use motion information as well as colour information (section 4.3) and have made the following contributions:

- Incorporating a motion probability map.
- Motion updates of the user strokes.
- Blur and distance fall-off parameters.
- Motion distance penalty.

Results using the new algorithm (figure 2-1 and section 4.5.1) show noticeable improvements when segmenting characters from stop-motion footage for both single frames and video.

## 2.2 Background

### 2.2.1 Production Background

Looking at some typical production schedules for commercials, during a 30 day production for a 20-second commercial, often 8 or 9 days are spent removing rigs and on post-production clean-up. This is greater than the 20% quoted above and a significant percentage of the production's duration. It highlights why there is a company need for research in this area.

To demonstrate the scale of the problem, Aardman Animations' stereoscopic film "Pirates: In an Adventure with Scientists" is 88 minutes long. This means that it has 126720 frames, as the film is shot at 12 frames per second and in stereoscopic format. The animation is shot in "doubles" which means that each frame is repeated twice. Not every one of the captured frames contains rigging, but the volume and complexity of the rigging equipment used in this film was substantial. For example, one shot contained eight characters each with their own rigging equipment. These rigs had to be removed in post-production along with the shadows each rig was casting on the scene. The rig removal for the film created so much work that some needed to be outsourced to studios such as Double Negative (Failes, 2012). If tools could be improved in-house then reducing or removing outsourcing would make a significant saving for the company. A segmentation tool would need to work effectively on production size images. Typical production frames are at least 1920 pixels in width by 1080 pixels in height and many productions have larger frame sizes.

The problem of rig removal is not specific to stop-motion animation and is prevalent in other areas of the industry. Rigging equipment is used in live-action footage in a similar way. For example, if an actor is doing something that is not physically possible, they may be held in place by some form of rigging or wires. Again these materials need to be extracted from the final frames and replaced with background content. Any tools developed could also extend to help with the extraction of other objects from footage.

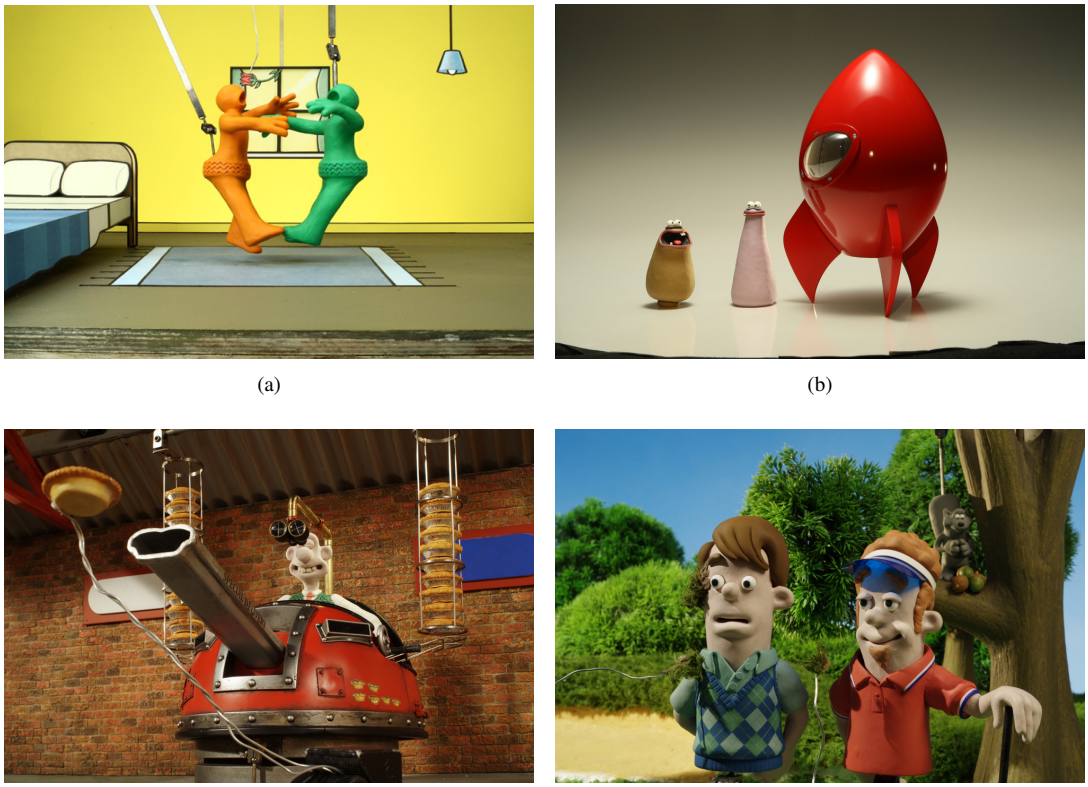
#### Rigging Equipment

The rigging equipment used in production can vary greatly (figure 2-3) and does not have a consistent form. The variety of objects used as rigging reflects the way that animators work and the solution must not interfere with this creative freedom. Table 2.1 categorizes typical rig removal work by rig type and scene type. The table uses these categories to estimate the complexity involved in removing the rig. Each situation is marked on a scale of 1 to 5, where 1 is the most simple to fix and 5 is the most complicated.

Table 2.1 shows that the rig removal jobs handled in post-production vary greatly in complexity dependent on the circumstances of each individual frame. The most simple rig removal jobs are those which have consistently coloured rigs on a plain, flat background, such as the rigging in figure 2-3a. For these jobs a mask can be quickly created in the key area of the join between the rig and the attached

| | | Rig type | | | | Comments |
|---|---|---|---|---|---|---|
| | | Thick metal rig | Thin metal rig | Plasticine rig | Cocktail stick | |
| **Scene type** | Rig in front of plain background (for example figures 2-3a and 2-3b). | 1 | 2 | 3 | 1 | This is the most simple scenario for rig removal. The reflections on the thin metal rigs may mean that mask adjustments by hand are required. The plasticine rig will require careful masking by hand if of a similar colour to the object. |
| | Rig in front of populated background (for example figure 2-3d). | 2 | 2 | 3 | 2 | The busy background would require the rig to be masked by hand as selection tools can struggle with finding the boundaries accurately. The key area for the mask being the join between the rigging and the attached object. |
| | Rig in front of moving background components. | 4 | 4 | 4 | 4 | The rig will need to be masked accurately by hand and the moving background components need to be filled, for which there may not be a clean plate available. |
| | Rig occludes moving foreground component (for example figure 2-3c). | 5 | 5 | 5 | 5 | The rigging must be masked very accurately by hand for the entire occluded area. The foreground component must then be filled, for example by using frames where the area is visible. |
| | Rig occludes attached object. | 5 | 5 | 5 | 5 | The rigging must be masked very accurately by hand over the occluding area. The attached object must then be filled using content from frames in which the occluded area is visible. |
| | Rig casts shadows on background components. | 2 | 2 | 2 | 2 | Background areas affected can be masked roughly by hand and replaced with the clean plate. |
| | Rig casts shadows on foreground components or attached object. | 5 | 5 | 5 | 5 | The shadows cast on the foreground components or attached object must be masked by hand. These areas must then be filled or adjusted to remove the shadow. |

Table 2.1: An analysis of the complexity of rig removal tasks depending on the type of rig used and the scenery types. The complexity in each case is rated from 1 to 5 with 1 being the most simple and 5 being the most complicated to remove and repair. Where ratings remain the same for all rig types, the type of rigging is immaterial to the work required.



(a) (b)

(c) (d)

Figure 2-3: Example frames showing different rigging equipment ©Aardman Animations Ltd 2014. (a) shows an example of rigging that is relatively simple to remove; metal armatures on a plain background. (b) shows a piece of plasticine used as a rig under a jumping character. This is a more difficult problem due to the similarities between the character and the rigging as they are made of the same material. (c) shows the use of a thin wire rig. Removal of this rig is a more complicated due to the rig occluding a foreground object, with the rig and occluded object both moving in the scene. (d) shows thin wire rigs in a busy scene. This is another example of complex rig removal due to the thin and reflective rigging, along with the complexities of matching the busy background over video frames.

object. This mask is moved as the object and rigging move and the area is then replaced with content from the clean plate. More complicated rig removal tasks involve rigging which occludes foreground objects, such as the rigging in figure 2-3c. This rig made of thin metal is not easy to mask as it moves through the frames. The mask must be accurate and the area must be carefully replaced using imagery of the same foreground object from other frames.

It has been decided by the company that it is not possible to dictate that rigging objects must look a specific way, for example by painting or adding patterns to the equipment. The processes used during stop-motion production are protected, as changes could interfere with the animators' workflow. When rigging a character, an animator will use whatever is nearby to get the character into position. For example, figure 2-3b shows a rig that is simply another piece of plasticine placed under the character whilst it is in a jumping position.

### Rotoscoping

Currently, rigging objects are removed from film frames using a method called rotoscoping, which is the process of manually changing video footage frame by frame (Seymour, 2011). There are different ways of rotoscoping objects in frames and a variety of software that can be used. There are existing tools available to reduce time spent rotoscoping, but it remains a time-consuming and labour-intensive job.

Rotoscoping is used for a number of different tasks in post-production including rig removal, stereo conversion and green-screen compositing. For the task of rig-removal the rigging object needs to be outlined on every frame in order to separate it from the character or object being held by the rig. If it is easier the character or object will be extracted instead of the rig. The most important area for accurate segmentation is the boundary between the rig and the object.

As well as a rotoscoping tool outlining the edges of the object, it needs to be able to follow this object as it moves between frames in the video clip. If the tool can work on multiple frames with minimal input then it saves the user a significant amount of time. Tracking the motion of an object can help the results to be temporally consistent over a number of frames.

The output of rotoscoping is a matte detailing the object area. A matte is a greyscale image in which the black areas will be opaque and the white areas transparent. Any grey-value will have some degree of transparency dependent on the colour depth. This matte layer, or alpha channel, is placed over the original image to remove unwanted areas.

There are a few different methods used to create a matte and these can be used to separate the rig from the object to which it is attached. The main techniques used are extraction, rotosplining, painting or a combination of the three.

- **Extraction** "Extraction is the process of procedurally generating a black and white matte." Seymour (2011). An example of extraction would be to use a colour keyer or a luminance keyer to extract a matte based on a particular colour or luminance value in the image. Extraction is the most automatic method of rotoscoping and the least labour-intensive. However, it is very dependent on the contents of the image and whether there is suitable information to get an automatic segmentation.

- **Rotosplining** "Rotosplining is the process of creating vector shapes to manually cut an element out of its background" Seymour (2011). The vector shapes are then moved as the object moved through the frames. The user must position key-frames and software can estimate the in-betweens.

- **Painting** Painting the mattes by hand is the most time-consuming form of rotoscoping. It is also not always effective as the results can look inconsistent across frames. This method is generally used as a last resort or to amend something small after extraction or rotosplining.

### Software and Tools

There are a range of different compositing and post-production tools on the market, such as After Effects (Adobe, 2012), Nuke (The Foundry, 2013b) and Flame (Autodesk, 2014). Although there are many options available, one of the key requirements defined by the company was that the final tools should

work within the company's existing production pipeline. Therefore, it was necessary to develop tools to work with the Nuke application and the Linux operating system.

### 2.2.2 Related Literature

There has been a lot of research into segmentation and interactive segmentation tools. It remains an active research area. Examples of popular segmentation methods include Mean-Shift (Paris and Durand, 2007), Berkeley segmentation (Arbelaez et al., 2009) and Graph Cut techniques (Kohli and Torr, 2007). Segmentation research can be split into two distinct categories, region-based methods and boundary-based methods. Examples of boundary-based methods include Intelligent Scissors (Mortensen and Barrett, 1995), Image Snapping (Gleicher, 1995) and Berkeley Segmentation (Arbelaez et al., 2011). Examples of region-based methods include Photoshop's Magic Wand, Graph Cut (Boykov and Jolly, 2001), Grab Cut (Rother et al., 2004) and Interactive Photomontage (Agarwala et al., 2004a).

Some methods categorize objects by training the system over time to recognise a particular type of object (Sivic et al., 2005; Shotton et al., 2005; Marszatek and Schmid, 2007). For example, if given a set of images of a car, the computer can use these images to learn what a car looks like and then look for car objects in subsequent images. The computer is trained to learn about the object and images over time (von Ahn et al., 2006). Using this approach for rig removal would depend on the range of rigs used in production and whether an effective training set could be built.

Other methods use additional information gathered from the scene during capture. A way of analysing an image is to use structured lighting (Kim et al., 2011). Shining patterned lights onto the scene can help to distinguish object boundaries. It has the additional benefit of providing some depth information about the scene, which can be used to determine the position of objects. Sun et al. (2007) uses lighting to detect foreground objects in a scene. In this case the segmentation is limited to being able to extract foreground objects. However, it shows how capturing an extra pass with different lighting can help to gather additional knowledge about the scene.

It has been stipulated in the requirements that any tool developed must not require changes to the traditional methods used on set. Therefore methods which require an animator to use a certain type of rig or change the environment on set are not feasible for this project. Due to these constraints, it has been decided that the tool will receive images, captured using the standard process, as input. Therefore, the tool will require some form of user interaction to indicate the areas to be segmented.

This project requires a method that incorporates user input to mark the object and background regions. Existing methods use a range of interaction types. Region-growing techniques require users to input at least the seeds of the object region and the background region. Some methods require user strokes on both the object and background regions in the image (Li and Shen, 2010; Boykov and Jolly, 2001; Levin et al., 2006). Graph cuts (Boykov and Jolly, 2001) use the input to segment the image into object and background regions using a 'max-flow' algorithm after assessing both the regional and boundary properties. GrabCut (Rother et al., 2004) builds upon graph cut techniques and reduces the amount of user interaction required to one bounding box. GrabCut uses colour data models in the form of Gaussian mixture models along with an iterative energy minimisation algorithm. Lazy Snapping (Li et al., 2004) provides an interactive cut-out tool for users to create a mask around an object by combin-

ing region-based and boundary-based cut-out techniques. It uses graph cuts (Boykov and Jolly, 2001) as the basis for the segmentation with the addition of poly-line editing.

Bai and Sapiro (2007) use geodesic distances to segment images. The underlying idea is to use a distance measure that analyses the image gradient as well as the physical distance along paths between a pixel and the object area. Using geodesic computation appears to be more efficient and less computationally expensive than graph cut techniques. The algorithm uses weighted distance functions based on the geodesic distance to the user's input. The results are good with accurate segmentations for relatively little user input. Geodesic segmentation was developed to produce accurate segmentation results for high resolution images and video and it claims to be up to 60 times more efficient than max-flow techniques (Criminisi et al., 2008). This improvement is explained by the addition of a restricted search space, contiguous memory access and parallel processing.

Video segmentation algorithms can require different user input to that of image segmentation. Interactive Video Cutout (Wang et al., 2005) allows users to indicate object regions over space and time. In contrast, a Mixture of Trees approach (Badrinarayanan et al., 2013b) proposes a solution for coping with the segmentation of fast shape changes and motion blur with user input only on the first frame. A method by Agarwala et al. (2004b) requires curves to be drawn on the first frame so that they can be tracked through subsequent frames. Other methods require a correct segmentation on the first frame of the sequence and propagate subsequent frames using that information (Bai et al., 2010, 2009). The information extracted from the user input also varies for different algorithms. A combination of cues such as motion, gradient, colour or shape (Bai et al., 2009; Price et al., 2009; Corrigan et al., 2008) can provide information about the object and background regions.

The development of video segmentation for use in the production community is an active area of research (Bai et al., 2009; Corrigan et al., 2008; Kokaram et al., 2005b,a). Successful state of the art research that has been implemented into production is that of Bai et al. (2009). The algorithm requires an initial segmentation as input and uses a combination of colour, texture and motion cues to track the segmented region over time. These cues are evaluated locally along the boundary areas, as well as globally across the whole frame. This research was developed to become the basis of the RotoBrush tool in Adobe After Effects (Adobe, 2012).

Geodesic distance has been used for video as well as image segmentation. Criminisi et al. (2010) introduce the Generalized Geodesic Distance Transform (GGDT) for images and video. This algorithm only requires a minimal user input of object and background strokes and it speeds up the segmentation process considerably. It removes the need for energy minimization and produces results comparable to graph cut techniques in a fraction of the time. This means that the algorithm can be used effectively on much larger image sizes, which is a key requirement for a production tool.

This research extends the state of the art geodesic segmentation method of Criminisi et al. (2010). The algorithm uses the simple object and background user interaction that the company require from a tool. However, an implementation of this algorithm failed when tested on production images. It was not possible to achieve an accurate segmentation when the object and background colour ranges were similar. Additionally the quality of video segmentation would degrade very quickly when the target object was moving, with holes in the segmented area appearing in subsequent frames. One reason for this might be the nature of stop-motion animation and how the motion differs from that of live-action

footage. As previously mentioned, stop-motion animation is characterised by small jerky movements rather than the smoother motions of live-action. My contribution is to improve segmentation on frames with limited colour ranges and to sustain the segmentation over a greater number of video frames.

Since this project, new methods of video segmentation have been published. Wang and Collomosse (2012) introduce an algorithm that uses a multi-label graph cut on frames, along with superpixels to encourage spatial coherence. Superpixels have also been used by other recent methods (Galasso et al., 2012; Dondera et al., 2014), where Galasso et al. (2012) use them to represent the most powerful similarities between video frames, along with motion cues to improve the segmentation performance. Papazoglou and Ferrari (2013) present a fully automatic video segmentation algorithm, which extracts foreground objects from video footage. A new semi-supervised video segmentation algorithm has been proposed by Badrinarayanan et al. (2013a), which uses a tree structure to link patches between frames in the video sequence.

Interaction methods have reduced to that of a single finger touch, as demonstrated by a new image and video segmentation algorithm by Wang et al. (2014). It performs the segmentation based on edge, texture and geometry information, at the location of the finger touch. To provide a range of accessible test footage for video segmentation algorithms, Galasso et al. (2013) have introduced a new benchmark dataset. This dataset provides manual segmentations completed by multiple users, to which video segmentation algorithms can be compared. Ochs et al. (2014) use point tracking over a large range of video frames to analyse motion cues. They claim improvements in segmentation results, for example successful motion segmentation even in frames where objects are stationary for periods of time. In addition to this new algorithm, they also add to the motion segmentation dataset introduced in Galasso et al. (2013).

## 2.3 Method

When studying existing examples of rigging in footage, it was noticed that the characters or objects attached to the rigs move a lot more coherently than the rigs themselves. The rigs can often switch to a different position in the scene during the sequence. Additionally rigging objects are often long and thin which is known to be a difficult shape for segmentation algorithms. For these reasons research was focussed on extracting the character as the way of separating the character from the rig. This was confirmed by a conversation with post-production in which they stated that they were equally happy with a segmentation of the character or a segmentation of the rig object, as the most time consuming task is to get an accurate segmentation between the two. Therefore interactive segmentation algorithms that could be applied to characters in stop-motion footage were researched. The investigations into finding a suitable benchmark tool for video segmentation of stop-motion animation are discussed in section 5.4.1.

It was found that a state of the art method (Criminisi et al., 2010) struggles with frames in which the object and background colour ranges are similar (shown in figure 2-4c) and degrades quickly when processing stop-motion video. This research proposes an improved solution by using motion information in conjunction with colour information. Both the method of Criminisi et al. (2010) and the new

algorithm presented in this chapter use geodesic distance transforms.

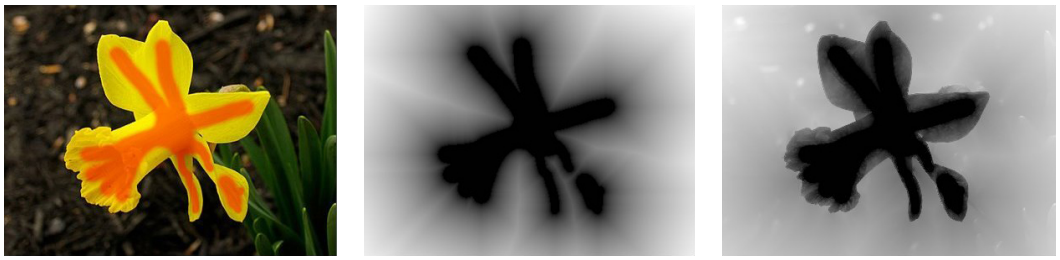
(a) (b) (c)

Figure 2-4: (a) The user input for the object region. (b) The user input for the background region. (c) Segmentation using the colour probability map (Criminisi et al., 2010). ©Aardman Animations Ltd 2014.

### 2.3.1 Theory of Geodesic Distance Transforms

A variety of geodesic distance transforms are calculated within the algorithm of Criminisi et al. (2010). This next section explains their geodesic segmentation algorithm and how the final geodesic distance transform used for segmentation is calculated.

**Geodesic Distance Transforms** The algorithm requires user input in the form of user strokes over the object to be segmented. The strokes on the object area are used as a mask to which the distance transform is applied. The distance transform measures the distance from every pixel in the image to the user-specified object area. The transform uses geodesic distances which assess the image gradient over the path as well as the Euclidean distance. This means that the distance will be greater if there are strong boundaries along the path from the pixel to the object area. Figure 2-5c shows how the geodesic distance highlights the boundaries of the object area.



(a) Original image with foreground input.

(b) A Euclidean distance transform based on the foreground input shown in (a).

(c) A geodesic distance transform based on the foreground input shown in (a).

Figure 2-5: The outputs of a Euclidean distance transform and a geodesic distance transform. The results shown in (c) demonstrate the use of boundary information in geodesic distance transforms. The distance values increase significantly when the path between a pixel and the foreground input strokes encounters strong object boundaries.

Criminisi et al. (2010) initially suggest equation 2.1 for the calculation of the distance transform with $d(a, b)$ computed by a raster scan approximation detailed in Toivanen (1996) (detailed in appendix A). This method traverses the image two times, from top-left to bottom-right and vice versa, to give the final minimum distance values from every pixel to the object area.

$$D(x; M_{col}, I) = min(d(x, x')) + \nu M_{col}(x) \tag{2.1}$$

where $x$ is the pixel position, $M_{col}$ is the colour probability map, $I$ is the input image, $\nu$ is a weighting factor and

$$d(a, b) = \inf_{\Gamma \in P_{a,b}} \int_0^{l(\Gamma)} \sqrt{1 + \gamma^2 (\nabla I(s) \cdot \Gamma'(s))^2}\, ds \tag{2.2}$$

where $P$ is the set of all paths between $a$ and $b$, $\Gamma(s)$ is one such path, parameterised by its arc length $s \in [0, l(\Gamma)]$ and $\gamma$ is a weighting factor typically set between 1 and 10.

The colour probability map, $M_{col}$, is generated using the pixel values from object and background user input (example foreground and background user inputs are shown in figure 2-4).

**Generalised Geodesic Distance Transforms** A Generalised Geodesic Distance Transform (GGDT) is developed in Criminisi et al. (2010). This transform uses user strokes over both the background and foreground areas to create probability histograms based on the pixel values in the strokes. The probability of each pixel belonging to the foreground or background is calculated and this generates a soft probability map which is used to initialise the distance transform. Equation 2.3 shows the inclusion of the generated colour probability map, $\nu M_{col}(x)$ into the geodesic distance transform.

$$D(x; I) = min(d(x, x')) \tag{2.3}$$

where $D(x; I)$ is initialised to be $\nu M_{col}(x)$

**Further Distance Transforms** The distance transforms are combined in a number of ways to create the final image segmentation. A Signed Generalised Geodesic Distance Transform (SGGDT), is calculated using a GGDT for the object area mask and a GGDT for the complement of this mask, as shown in equation 2.4.

$$D_s(x; M, \nabla I) = D(x; M, \nabla I) - D(x; 1 - M, \nabla I) \tag{2.4}$$

The Symmetric Signed Generalised Geodesic Distance Transform (SSGGDT) (equation 2.5) is calculated by performing geodesic erosion and dilation (equations 2.6 and 2.7). This stage helps to refine the edges of the segmentation at the object boundary and remove areas of noise.

$$D_S^S(x; M, \nabla I) = D(x, M_e, \nabla I) - D(x; \overline{M_d}, \nabla I) + \theta_d - \theta_e \tag{2.5}$$

where

$$M_e(x) = [D_s(x; M, \nabla I) > -\theta_e] \tag{2.6}$$

$$M_d(x) = [D_s(x; M, \nabla I) > \theta_d] \tag{2.7}$$

and $\theta_e$ and $\theta_d$ are a user settable variable.

The final step is to filter the SSGGDT using Geodesic Symmetric Filtering. This is simply a case of thresholding the image at zero to output the final segmentation (equation 2.8).

$$M_{GSF}(x; M, \nabla I) = \left[D_S^S(x; M, \nabla I) > 0\right] \tag{2.8}$$

**Video** The distance transforms detailed are used to segment individual images. Criminisi et al. (2010) also state that this algorithm segments video content accurately and with temporal consistency. They claim that geodesic segmentation can be easily extended to video by adding a dimension to create a video cube. The cube represents the x and y values from the frame with the addition of the third dimension of time. The same processing performed on the 2-dimensional image can be extended to 3-dimensions. However, Criminisi et al. (2010) do not detail the exact scanning methods used for a 3-dimensional video cube.

## 2.4 Improved Image Segmentation using Motion

The method of Criminisi et al. (2010) was implemented into the production systems at Aardman Animations. However, when tested on production footage, its reliance on colour information led to inaccuracies in the segmentation results. For this reason a new algorithm was developed, which incorporated motion information as well as colour information. The details and contribution of my new algorithm are described in this section. It is followed by results, comparing the state of the art method of segmentation algorithm (Criminisi et al., 2010) to the new algorithm (section 4.5.1).

### 2.4.1 Incorporating a Motion Probability Map

It is known that the object to be segmented will be moving, so motion cues used in conjunction with colour cues provide valuable information about the boundaries of the target object. My new method uses equation 2.3 to calculate geodesic distances. However, it is initialised using a probability map based on both motion and colour information, rather than colour alone. So $D(x; I)$ is initialised to be

$$D(x; I) = \nu(M_{col}(x) + \lambda M_{mo}(x)) \tag{2.9}$$

where $\lambda$ is a weighting factor for the strength of the motion probabilities (typically $\lambda$ = 0.5 is used).

The motion probability map is generated using motion flow values. Forward and backward motion flow is calculated for every pixel using Vector Generator (The Foundry, 2013a). One frame's forward vectors are shown in figure 2-6 with the positive u and v values represented on the red and green channels. The forward and backward motion values are averaged so that information from both directions is included and so that each pixel has one velocity vector which is stored as u and v coordinates in a motion map $M_{mo}(x)$.

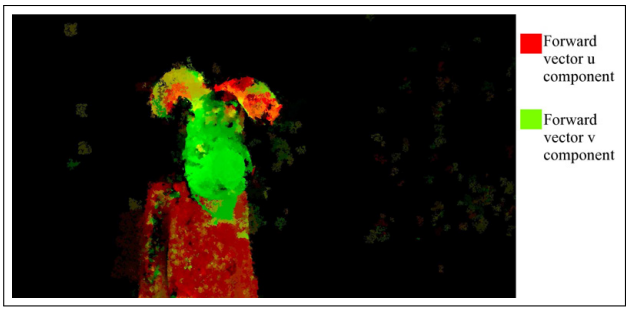



Figure 2-6: An example of forward motion vectors in one frame, calculated using Vector Generator (The Foundry, 2013a) ©Aardman Animations Ltd 2014.

Figure 2-7: Segmentation using the motion probability map ©Aardman Animations Ltd 2014.

Pixels in the object or background user strokes populate object or background motion histograms with the pixel's u and v values. The values are bucketed to keep like motions together. This algorithm uses 32 buckets for each axis in a similar way to the colour buckets defined in Criminisi et al. (2010). The motion probability map $M_{mo}$ is calculated using the object and background motion histograms. The calculation of the map is the same as that of the colour probability map in Criminisi et al. (2010) and is shown by equation 2.10.

$$M_{mo}(x) = \sigma\left(ln\frac{p(z(x)|\alpha(x) = Bg)}{p(z(x)|\alpha(x) = Obj)}\right) \tag{2.10}$$

with $\sigma(.)$ the sigmoid transformation $\frac{1}{\sigma(t)} = 1 + e^{-t/\mu}$ and where $z(x)$ is the pixel at position $x$ in the input image, $\alpha$ is the initial binary segmentation at pixel $x$ and $\mu$ is variable (I use $\mu = 5$).

As shown by figure 2-7 the motion map alone does not provide an accurate mask. Therefore, my algorithm uses the colour and motion probability maps together to initialise equation 2.3. This change to the algorithm affects the subsequent calculations in Criminisi et al. (2010). That is the signed GGDT, signed symmetric GGDT and the geodesic symmetric filter are all updated to now analyse both colour and motion cues.

Figure 2-8: The original user strokes for frame 1 are shown in blue. The algorithm's updates of the user strokes by frame 3 is overlaid in red. The image content shown is that of frame 3, showing the updated strokes moving in line with the character's movement. ©Aardman Animations Ltd 2014.

### 2.4.2 Motion Updates of User Strokes

The motion information generated is further used to improve the quality of the video segmentation. The segmentation algorithm of Criminisi et al. (2010) deteriorated quickly on stop-motion footage containing a target object in motion. The velocity vectors obtained for the motion probability map are used to update the user input strokes (figure 2-8). The input on the initial frame is moved to subsequent frames by the amount specified in the pixel's velocity vector. For example, if on frame $t$ a user has indicated an object pixel at position $x$, then frame $t + 1$ will have an object pixel at $x + M_{mo}(x)$.

### 2.4.3 Blur and Distance Fall-Off Parameters

Additional user controls have been added to the tool to improve the final output. I found that blurring the initial probability map helps to remove false seeds in the initialisation map, i.e. small groups of pixels that are outside of the object area and would be falsely identified as part of the object due to similarities in colour and/or motion. Users are provided with a blur control to remove these unwanted areas and can vary the strength of the blur.

Figure 2-9: Segmentation using the colour probability map modified with blur and distance fall-off parameters ©Aardman Animations Ltd 2014.

A distance fall-off control has also been added to the probability map to prevent distant areas of

similar colour or motion from being segmented. The fall-off is calculated by finding the Euclidean distance from each pixel to the object area. This distance value is scaled and added to the probability map values so the further the pixel is from the object area, the greater the penalty added. Figure 2-9 shows the improvement with these added controls.

### 2.4.4 Motion Distance Penalty

A further addition to the algorithm is to use the motion data within the distance calculation as a penalty. The distance transform used in Criminisi et al. (2010) is shown in equation 2.2 and is approximated with a raster-scan method (Toivanen, 1996). I have updated the algorithm so that a user can opt to penalise against strong changes in motion as well as strong image gradients. The integrand of equation 2.2 is replaced with that in equation 2.11, causing changes in the motion vectors along the path to be analysed as well as the changes in colour. The weight of this penalty, $\rho$, is added as a user control so that users can vary its strength depending on the content in the footage.

$$\sqrt{1+\gamma^2((\nabla I(s)\cdot\Gamma'(s))+\rho(\nabla M_{mo}(s)\cdot\Gamma'(s)))^2} \tag{2.11}$$

where $\rho$ is a weighting factor and is typically set between 0 and 1.

### 2.4.5 Implementation Details

This work has been implemented in C++ as a Nuke plug-in (The Foundry, 2013b) to allow easy integration into the existing production pipeline at Aardman Animations. Appendix B provides the source code to the main Nuke class for the plugin implementation. To improve processing times some of the calculations could be completed in pre-processing. The motion vectors and divergence values can be calculated as data is transported to post-production, leaving only the motion histograms and per pixel probabilities to be calculated at run-time once the user has entered the object and background strokes.

Figure 2-10: Segmentation using the final algorithm and optimised user controls. ©Aardman Animations Ltd 2014.

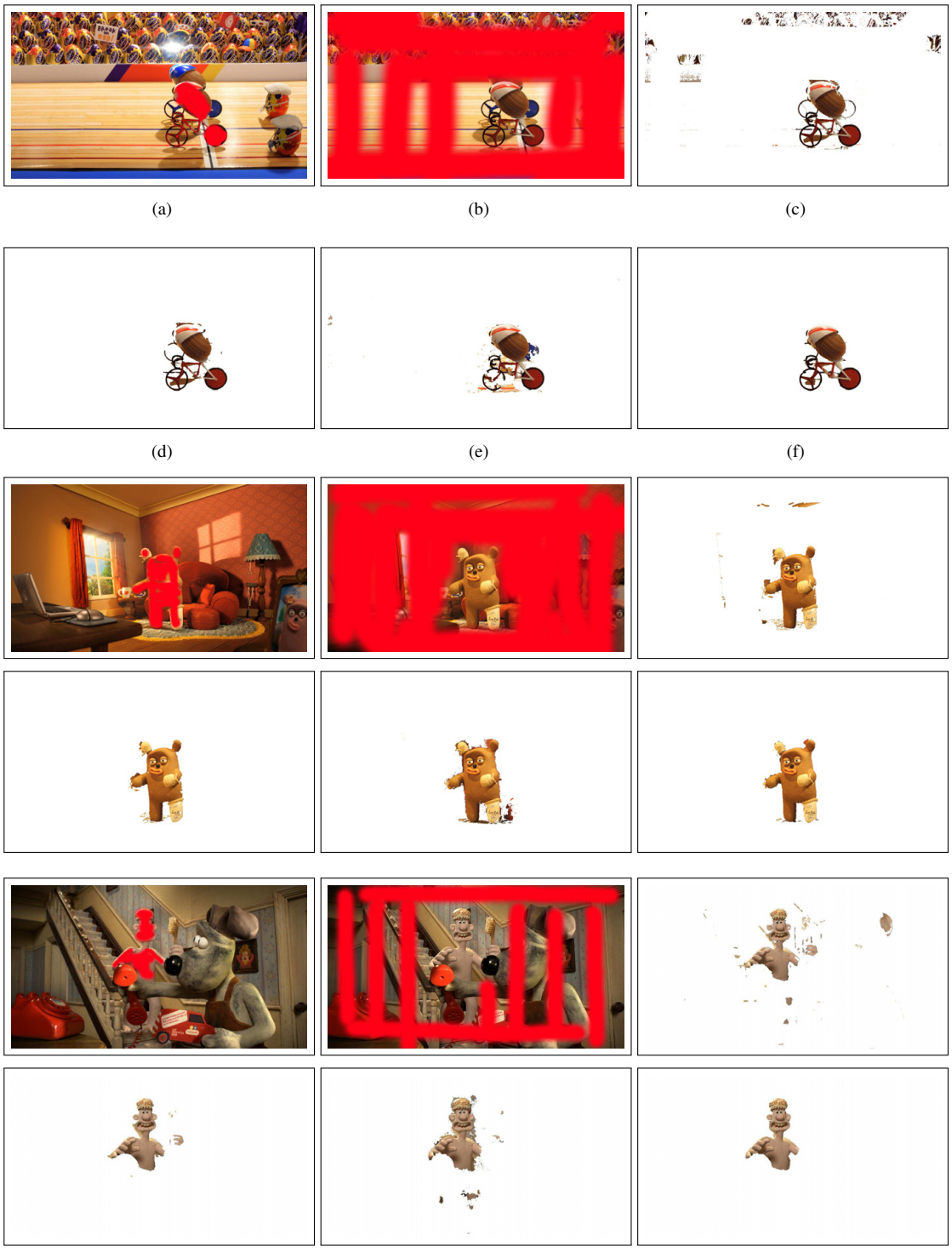



Figure 2-11: Results for individual frames. (a) The user input for the object region. (b) The user input for the background region. (c) Segmentation using the colour probability map (Criminisi et al., 2010). (d) Segmentation using the colour probability map modified with blur and distance fall-off parameters. (e) Segmentation using the motion probability map. (f) My results using a combination of motion and colour information. Images in the first example ©Cadbury Creme Egg, the second example ©National Accident Helpline and the third example ©NPower.

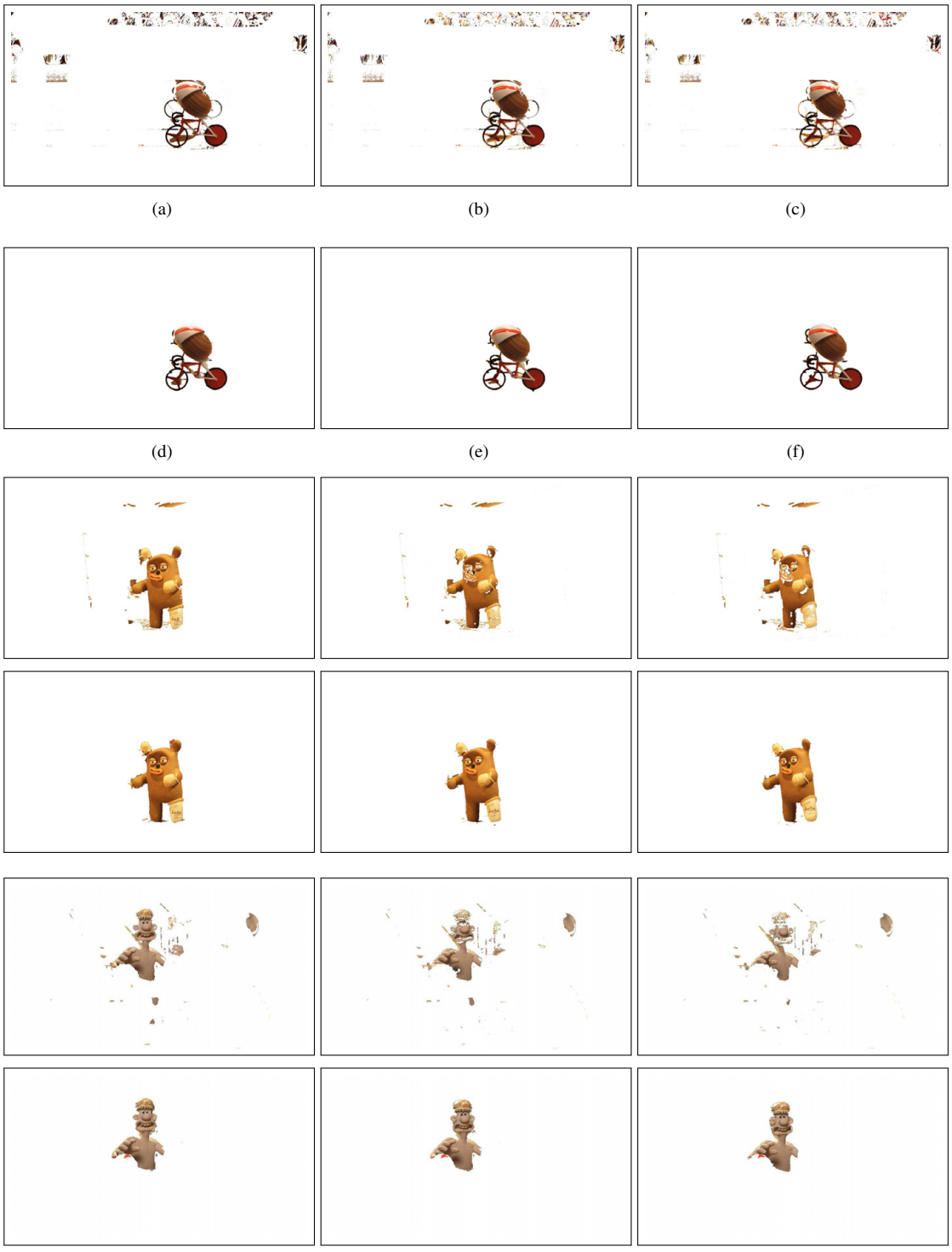



Figure 2-12: Results for video. (a) - (c) Frames 1-3 of the video segmentation using a state of the art algorithm (Criminisi et al., 2010) based on colour information. (d) - (f) The same frames using my segmentation algorithm which is based on motion and colour information. Images in the first example ©Cadbury Creme Egg, the second example ©National Accident Helpline and the third example ©NPower.

## 2.5 Experiments

To test the new algorithm it is compared to a Nuke implementation of the state of the art segmentation algorithm (Criminisi et al., 2010). Both implementations are run on a number of production images containing characters that Aardman Animations would like to be segmented. The images chosen are examples of production images for which the original captured footage would be likely to contain rigging to help the character move. Therefore these expeirments are designed to test the extraction of the character object to see whether it could be accurately separated from a rig. The input user strokes, indicating the object and background regions, are identical for each algorithm. Each algorithm's parameters are optimised to generate the most accurate segmentation results.

I first test the algorithms on individual video frames from commercial footage. The inputs, the results from the comparison algorithm (Criminisi et al., 2010), the contributions of my research and the final results using my algorithm are shown in figure 2-11. I then test my implementation on multiple frames to assess how it compares for video footage (figure 2-12). The results for the implementation of Criminisi et al. (2010) and my implementation are shown over three frames of stop-motion footage in which the character to be segmented is moving.

### 2.5.1 Results

The results of the algorithm presented show a noticeable improvement in segmentation when compared to those of Criminisi et al. (2010) (figures 2-11 and 2-12). They perform a more accurate segmentation for both individual production frames as well as video footage. When the colour probability is used alone, the results show some segmented areas unrelated to the target object because they share colours with the target area. The new method incorporating motion and colour information removes many of these errors in the segmentation.

Figure 2-11 shows results generated on individual video frames. The foreground and background input were rough strokes drawn by the user. The precise position of the foreground and background strokes within the object and background areas do not significantly affect the resulting segmentation. However, it is important to have sufficient strokes in each area and, most importantly, that all the colours contained in the foreground and background areas are well represented. Fewer marked pixels, as foreground in particular, reduces the quality of the segmentation result. It is essential to make sure that areas narrowly connected with the main object contain user strokes, such as the character's ears in the second example. In future work it would be worthwhile to repeat the experiments with different markings, calculating the percentage of marked pixels and how that relates to the quality of the final segmentation. The results could be used to provide a detailed recommendation to users of the algorithm and suggest the best practice for achieving the highest quality segmentation results.

## 2.6 Discussion

### 2.6.1 Individual Frame Results

Figure 2-11 shows the results on individual production frames. There is a clear improvement in using motion cues in conjunction with colour cues (f) in comparison to the state of the art method (Criminisi et al., 2010) which uses only colour cues (c). User strokes indicate areas of object (a) and background (b) in the image. Using this same input for all methods, the results show the differences between state of the art and my additions to the algorithm. Image (d) shows the improvement of adding blur and fall-off parameters to the original probability map. Image (e) shows the results when swapping the colour probability map for the motion probability map. The motion cues alone do not provide good results because the motion analysis is not accurate enough. However, when the motion cues are combined with the colour cues (f) the results are much improved and demonstrate the best results.

The new method presented here significantly reduces the number of segmented areas found away from the object and improves the accuracy of the mask around the character. However, although the segmentation error has been reduced, the results still show some incorrectly segmented areas. These are generally pixels adjacent to the segmented object which contain the same colour values as pixels in the object area. To obtain the segmentation results required in production, these areas would need to be refined by hand after the segmentation tool has been run. Reducing the amount of clean-up is one area which requires future development.

Figure 2-13: An example failure case showing segmentation errors around Wallace's hands ©Aardman Animations Ltd 2014.

An example failure case of this new algorithm is shown in figure 2-13. The segmentation begins to error around the area of the character's hands and arms. The cause for this loss in quality is the combination of inaccurate motion information and similar colour values in parts of the background and foreground regions.

The forward motion vectors for this frame (depicted in figure 2-14) show that some of the static background pixels have been assigned motion vector values, as well as moving areas such as the hands and arms. These inaccuracies could be due to background pixels becoming exposed where they were occluded in previous frames. The erroneous motion data impacts on the quality of the final segmentation. To improve results in this type of situation I could either test the algorithm using an alternative

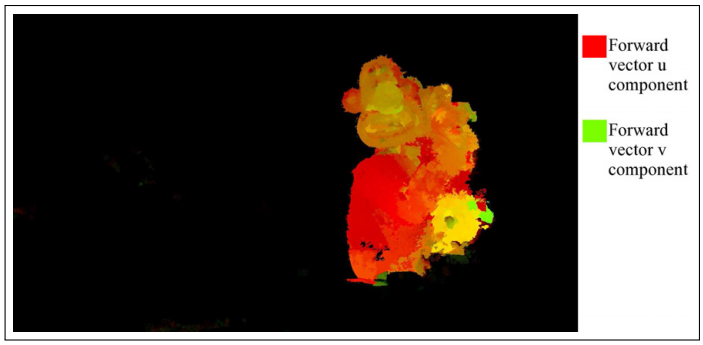



Figure 2-14: The forward motion vectors for the frame shown in figure 2-13, calculated using Vector Generator (The Foundry, 2013a) ©Aardman Animations Ltd 2014.

method of motion vector generation or make the algorithm less reliant on the motion information, for example, by introducing further analysis such as that of textures.

### 2.6.2 Video Results

Figure 2-12 shows results using the new algorithm ((d) to (f)) over multiple frames in comparison to my implementation of the algorithm of Criminisi et al. (2010) ((a) to (c)). To view a video segmentation result in motion, there is example footage contained in the supplementary materials of this portfolio. The inputs are identical for both algorithms with the user strokes shown in figures 2-11a and 2-11b used as input on the initial frame of the series. The results demonstrate a significant improvement. The algorithm of Criminisi et al. (2010) begins to degrade very quickly after the initial frame. However, my combined use of colour and motion maps, along with the movement of the user strokes over time, has provided notably more accurate results over multiple frames.

(a) Close up of the character object from frame 3 using the video segmentation algorithm of Criminisi et al. (2010).

(b) Close up of the character object from frame 3 using the video segmentation algorithm presented in this research.

Figure 2-15: Close up of results presented in example 2 of figure 2-12. ©National Accident Helpline.

(a) Close up of the character object from frame 3 using the video segmentation algorithm of Criminisi et al. (2010).

(b) Close up of the character object from frame 3 using the video segmentation algorithm presented in this research.

Figure 2-16: Close up of results presented in example 1 of figure 2-12. ©Cadbury Creme Egg.

Figures 2-15 and 2-16 show the results of character segmentation algorithms after 3 frames. Subfigures (a) show the results using the algorithm of Criminisi et al. (2010) and subfigures (b) show the results using the algorithm presented in this research. In figure 2-15a it is clear to see the deterioration over three frames when using the algorithm of Criminisi et al. (2010), with many areas of darker colours, such as the facial features and writing on the bandage, incorrectly assigned to background. Without the updating of user strokes the representation of these small areas of colour in the foreground is lost. By contrast, the results of my algorithm show much less deterioration, with the majority of the character segmented successfully. Some errors are visible around the left arm, where the background colour is similar to colours in the character's body and the motion of the arm has uncovered new areas of background, creating incorrect motion estimation.

Figure 2-16 shows an example segmentation result where the desired output was a segmentation of the foreground egg and bicycle. Figure 2-16a shows less deterioration in the algorithm of Criminisi et al. (2010) than that of 2-15a, with some loss of precision occurring at the edges of the character. The bicycle behind the selected character has been incorrectly included in the segmentation in places due to the similar colour ranges for the two objects. Figure 2-16b shows sharper, cleaner edges than that of 2-16a with very little deterioration of the segmented character. However, the handlebars of the bicycle behind have become segmented with the main object. This is due to the similarity in colour with the main bicycle object, the connectivity with the character to be segmented and the background bicycle being in motion on frame 3. This combination of colour and motion similarity in conjunction with the connection to the main object makes the background handlebars a difficult area to segment correctly. The deterioration present in the results of Criminisi et al. (2010) has meant that the segmentation of these handlebars is reduced in 2-16a, but at the expense of the cyclist's helmet which has also deteriorated around the edges.

### 2.6.3 Evaluation of segmentation results

An area for future work would be to formally evaluate the segmentation results. The evaluation would require a metric to describe the accuracy of the segmentation in comparison to a 'perfect' result. The ideal result could be represented by ground truth information for frames in the video footage. Ground truth information is available for a range of video footage in datasets such as the Video Segmentation Benchmark (VSB100) of Galasso et al. (2013) and the Freiburg-Berkeley Motion Segmentation dataset (FBMS-59) of Ochs et al. (2014).

The VSB100 dataset (Galasso et al., 2013) provides ground truths for 100 high definition videos from the Berkeley video dataset (Sundberg et al., 2011), the dataset includes footage representing a range of the challenges associated with video segmentation, such as occlusion and non-rigid objects. Multiple frames in the video sequences are annotated by four different people and the dataset comes with evaluation software. The annotations are used in evaluation to allow for ambiguity of correct segmentations by measuring natural ambiguity in the human segmentations. The evaluation software is designed to penalise temporally inconsistent segmentations.

The Berkeley Motion Segmentation Dataset (BMS-26) (Brox and Malik, 2010) is designed for the sub-problem of motion segmentation. The FBMS-59 dataset (Ochs et al., 2014) is an extension of the dataset of Brox and Malik (2010) and contains additional video footage. Both datasets provide ground truth segmentation and annotation for moving objects in the video frames. The dataset of Ochs et al. (2014) provides long and short sequences at different resolutions and includes challenging examples such as occlusion, multiple objects and changes in lighting. Ground truth is provided for every 20th frame in each sequence. Both the BMS-26 and the FBMS-59 datasets come with evaluation software, with the more recent FBMS-59 dataset of Ochs et al. (2014) claiming improvements to the metrics and the evaluation protocol. The new evaluation protocol allows for comparison of segmentations with different numbers of regions. They define a metric that takes into account accuracy and coverage of ground truth, by using precision and recall for segmentation.

It would be worthwhile testing the algorithm presented in my research on motion segmentation examples from the FBMS-59 dataset. It would mean that the algorithm could be formally evaluated and direct comparisons could be made with other video segmentation algorithms. The drawback of evaluating the algorithm in this way is that it would not be tested on the footage for which it was intended. The algorithm presented in this research was designed for the specific problem of segmenting moving objects in stop-motion footage. Therefore, it would not be a truly accurate evaluation to test it on other types of footage. For example, live action footage might contain motion blur and one assumption made when developing this algorithm was that motion blur would not be present.

In order to test the algorithm formally on the data for which it is intended it would be informative to build a dataset of stop-motion footage examples along with ground truth segmentations and annotations. The ground truth could be acquired by tasking post-production artists to carry out the segmentation of selected objects contained in frames. Once segmentations had been acquired from a number of different artists it would be valuable to review the amount of variation in their final segmented objects. Alternatively, there is the possibility of obtaining ground truth from previously completed production frames, using the masks created to remove rigs. However, these masks would probably not contain

accurate segmentations of whole objects and only important boundaries, such as the join between a rig and a character, would be defined accurately. The rest of the mask would be roughly completed and would therefore not contain a definitive ground truth. For this reason it is likely that the ground truth segmentation and labelling would need to be a separate task.

For a stop-motion video dataset, consideration needs to be given to which objects require ground truth segmentation and labelling. For the purposes of rig removal research it would be useful for the footage samples to contain scenes which use rigging equipment. The annotated frames should label the rig, the object attached to the rig and label the rest as background. If the rig was not present on certain frames then labels for the moving object and background would only be required. Although the rigging object is not required to test this particular algorithm, it would be useful to have it labelled as a benchmark for future segmentation algorithms. A full range of rigging examples, as described by table 2.1, would need to be represented in the footage in the dataset.

### 2.6.4 Recommendations for Rig Removal

The reason for this research was to help with the problem of rig removal. Table 2.2 analyses typical examples of rig removal and comments on how the proposed algorithm is likely perform when extracting characters or objects in a range of situations. It shows that there are a number of cases in which the proposed algorithm would help with rig removal. For example, common rig removal situations such as characters in front of stationary backgrounds, whether plain or busy, have been demonstrated to work well using the new tool.

There are other situations which are more complicated for the algorithm to process accurately, such as when there is movement from multiple objects in the scene. This movement might be from objects or from moving shadows and it can cause the motion data to be less reliable. If the motion data is unreliable and the colour data is similar between foreground and background areas then the algorithm is more likely to show erroneous areas. Rigs occluding areas of foreground also create additional work in post-production, but this work relates more to the job of replacement than the job of extracting the rig.

Having analysed a range rigging situations and how they each affect the processes required in post-production, some recommendations can be made to animators on set or shot designers. In order to reduce the work required on frames in post-production, I suggest that the following points are considered as best practice when rigging is required on set:

- Try to position the rig so that it does not occlude objects which are not present on the clean plate. This will make the job of replacing the rigged area much easier.
- Where possible, position rigs in front of non-moving parts of the scene. This allows the motion data to be used most effectively. There is flexibility in the positioning of rigs as they do not affect the final composition, whereas the objects attached to rigs must be placed precisely according to the shot design.
- Avoid using the same material for the rigging as the material of the object attached to the rig.
- Where possible, avoid shadows from rigging equipment being cast on foreground components in the scene.

| Type of scene | Complexity of extracting objects attached to rigging using the proposed algorithm |
|---|---|
| Plain background behind rig and attached object, for example 2-3a. | The proposed algorithm should be able to extract the attached object well and therefore separate it from the rigging object which can be replaced with the clean plate background. Results may be worse in the case that the rig is made of the same material as the attached object, for example 2-3b |
| Populated background (for example figure 2-3d). | The proposed algorithm should be able to extract the attached object. However, there could be errors in areas where there are strong similarities in colour along with inaccurate motion information. |
| Adjacent background or foreground components in motion. | The proposed algorithm may output erroneous areas where the attached object shows similarities in both colour and motion. |
| Rig occludes some foreground components (for example figure 2-3c). | The attached object could be extracted (dependent on the conditions above). Additional work would be required to replace the occluded areas of the foreground components. |
| Rig occludes attached object. | The attached object could be extracted but not in full due to the occlusion. Replacement work would be required to replace the content in the gaps. |
| Rig shadow cast on foreground components or attached object. | Extracting the foreground object or attached object would not help with removing shadows. A separate process would be needed to fix the shadows cast from the rig. |

Table 2.2: Analysis of the complexities of object segmentation, based on the type of scene. The 'attached object' is the object that the rigging equipment is holding and 'foreground components' are any other objects in the scene which will not be in the clean plate.

- Using rigs made of matt materials will help to reduce the number of colours in the rigging pixels, which form part of the background histogram. Fewer colours in the rig would make it more probable that the entire rigging object would be included in the background segmentation.

## 2.7 Conclusions

The new algorithm presented here shows an improved method of segmentation which takes advantage of the motion cues available in video frames. The algorithm is an extension of the work by Criminisi et al. (2010) which was found to rely heavily on colour cues and deteriorated quickly when tested on Aardman Animations' stop-motion video footage. The reliance on colour meant that it only worked well for frames in which the object and background colour ranges were significantly different. I found that this caused problems with much of the production footage, where the colour distributions are not always so distinct.

Motion is a very strong cue and I use this in conjunction with the colour information to improve the

accuracy of the results. The additional motion information was used to analyse the movement of the object which improves the accuracy of the method over multiple video frames. I found that the colour information and motion information complemented each other to provide an improved segmentation.

In future work it would be interesting to extend the implementation to become an automatically adaptive model. So by analysing the distributions of the motion and colour histograms I could make an assessment of the value of the data and set the weighting of $\lambda$ automatically. Similar object and background distributions, whether that be for colour or motion, are less useful than those in which there is a noticeable separation. So distinct distributions could be automatically weighted more strongly instead of the user manually adjusting the $\lambda$ variable.

Furthermore, additional data could be analysed to give insight into different properties of the object and background regions. I have considered looking at the texture information available in the user strokes and analysing it in a similar way to the colour and motion information. Building object and background histograms of texture data could be used in conjunction with the colour and motion models to add further accuracy to the segmentation algorithm. Analysing more information could help to correct inaccurate motion information such as motion vectors generated by occlusion or shadows. It could help to reduce any clean-up required after using the segmentation tool.

One area which would be key to such a tool becoming production useable would be to incorporate alpha matting around the edges of the segmentation. At present, all segmentation results have hard edges, but post-production require softer edges on their final output. It would be interesting to research intelligent alpha matting that could complement this segmentation work. Analysis of how users currently select the hardness of the edge would have to be completed to see if an algorithm could be used to produce a more user friendly final segmentation.

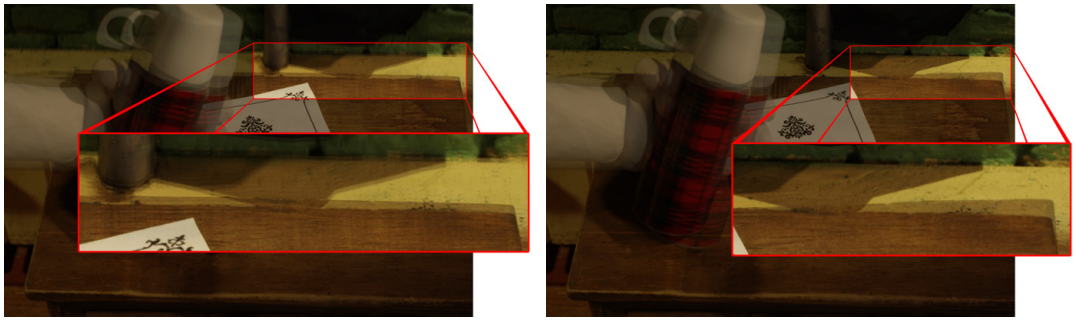
There is the potential for further research exploring solutions to the problem of rig removal. However, it was decided by the company that there were greater research priorities at this stage. Having looked at the results using state of the art research and the new algorithm on production images, the company felt that the speed and accuracy requirements were too great to be able to develop a production quality tool within the timescales and resources available. In addition to this, the tools available from off-the-shelf solutions had improved over the time of the research project, so the need for a tool was not as great as it had been at the beginning of the project. Therefore, other potential computational tools, which could help with the production of stop motion content, were investigated. After some analysis, it was decided that the next stop-motion problem to be researched was that of set shift.

# Chapter 3

# Set Shift

## 3.1 Introduction

Set shift is a problem in stop-motion production which occurs when items in a scene move unintentionally over time. The movements tend to be gradual and very subtle, such as a wooden board warping, and can not be noticed when capturing the footage. However, these movements become obvious when frames are played back in post-production and the movement is known as 'set shift' or 'set warp'. In this document it will be referred to as set shift. Set shifts result in small jumpy movements visible in the background of captured footage and are distracting to the viewer. Therefore, any shifts which occur during the capture stage must be corrected in post-production.



(a) A scenario where the background has shifted slightly.

(b) A scenario where the edge of a wooden table has warped.

Figure 3-1: Set shift examples showing the source image overlaying the target image. The highlighted areas show the set shift motion, by contrast the foreground arm is animated to be moving. The shifts have been exaggerated by increasing the motion of the shift for demonstration purposes, typically the shifts are very small, such as one or two pixels on a image of 2132 x 1421 pixels. ©Aardman Animations Ltd 2014.

Post-production artists have to correct the shifted object or objects in every frame in which they is visible, as well as any subsequent frames which show the item in its shifted position. This is a time-

consuming and labour-intensive process as it involves carefully outlining shifted objects and moving the masked area over time. This mask is used to remove the shifted objects and replace them with their original, correct positions.

Aardman Animations wish to investigate whether it is possible to develop a tool that could be used to correct shifted items automatically. The tool would require a source image, containing the shifted content, and a target image, containing the same area but without the shift. This chapter documents the research completed to assess the feasibility of such a tool. The problem was analysed in detail and research was completed into areas associated with the task, such as feature detection, tracking and image warping (section 3.2). Based on this research a prototype application was developed (section 3.3), which tests whether developing a solution to set shift is possible given the state of the art in the contributing areas of research.

The developed prototype was tested on example set shift images and the results are discussed in section 3.4. A number of difficulties were found in developing a tool that could cope with common examples of set shift and therefore a number of variations to the prototype were tested. These findings are presented along with conclusions on the feasibility of developing a set shift tool (section 3.5).

## 3.2 Background

### 3.2.1 Production Background

In order to understand the task of developing a tool to fix set shift thoroughly, some time was spent understanding how set shift is currently fixed in post-production. There are three situations that artists listed as common set shifts which require fixing:

1. There is a shifted area that is easy to isolate. The artist will create a mask around the affected area, and replace the content with that from a correct frame. The mask will be created with soft edges to ensure that no joins are visible.

2. There is a combination of moving items, some of which are correct movements and some of which are set shifts. This scenario becomes a more time-consuming problem. The areas that are moving correctly have to be rotoscoped more accurately so that fixes do not affect these areas.

3. There is a coherent shift across the scene, such as a small camera movement. Artists would use a tracker tool to stabilise the footage.

Both the first and third scenarios are relatively quick to fix. It is the second example for which users require an improved tool. Therefore, it must be assumed that there could be multiple moving objects in the scene with certain movements that need to be retained and certain movements, that is the set shift movements, which need to be replaced.

There are a number of example situations that artists listed as being particularly time-consuming, such as:

- If there are multiple elements in the scene which shift or warp differently. For example if a set is left over the weekend various elements may have moved by the next week when shooting

recommences.

- If there are shadows which are moving across a set shift area, as this means that the shift has to be fixed whilst retaining the movement of the shadows.
- When there is an object animated to be moving next to a shifted object, as the shifted area has to be very accurately masked to separate the two movements. It can mean that there is no 'correct frame' to copy the content from.

There are several tools which can be used in production to help with correcting set shift problems. After Effects CC (Adobe, 2012) offers the Warp Stabiliser VFX for warping footage. In Nuke (The Foundry, 2013b) combinations of the Spline Warp tool, the Tracker and the Planar Tracker can help to repair set shifts. However, more basic tools available in all post-production software, such as rotoscoping and replacement of image content, are often the best and most commonly used tools.

Users in post-production require this tool to be very accurate. The artists' current methods produce very precise and high quality results. It is essential that a tool will produce results to this same high standard when used by an experienced artist. It is expected that a post-production tool would improve on the time taken to replace an area with the results being of the same quality. If the tool requires a lot of tweaking in order for it to produce the same level of result then it will not be used.

As previous research has been completed into image and video segmentation during the rig removal project, it has been stipulated that research into these areas is not required. It was discovered during the rig removal project that segmentation is not at the level required for accurately extracting objects from production footage, which means that it will not be accurate enough for use in a solution to set shift.

### 3.2.2 Related Literature

The specific problem of set shift has not been addressed in research literature. However, there are several research areas related to this problem which require investigating. These areas include feature detection, edge detection, image warping and morphing.

As the input for this tool is two similar images, a shifted image and a correct image, the algorithm must include a way to match between them. The 'shifted image' will contain the frame showing footage containing a set shift of some description. The 'correct image' will contain the frame with the elements in the correct position, for example this might be a frame from before the shift occurred. In order to find the differences between the shifted image and the correct image, salient points or edges in each image need to be detected. Feature point detection or edge detection can be used on both images and then the results compared to find matches.

#### Feature Detection

There are a number of features which can be used for describing object positions in an image. Point-like features, such as corners can be detected by the Moravec (1980) corner detection algorithm. This algorithm was improved by the Harris corner detector (Harris and Stephens, 1988). Schmid and Mohr (1997) extended this method further employing a more general feature matching algorithm, rather than

local feature matching. Their features can be matched from a database of images by using a locally invariant descriptor of the region around the feature point.

Feature point detection developed from corners to any features of interest in an image, such as areas of local intensity maximum or minimum. Lowe (1999) introduces SIFT features, which are a popular and well-known method of feature detection. This method generates features which are invariant to scaling, translation and rotation. These feature points are filtered to find the most stable and distinctive keys. The keys contain information about the image gradients local to that area of the image and can be used to assess the match of features between images. SIFT features are shown to be very distinctive which means that a correct match can be found even in a large database of feature points (Lowe, 2004). Lowe (2004) presents a robust method for matching feature points which allows points to be correctly matched even when there has been transformations such as affine distortions, changes in illumination or changes in viewpoint.

Building on the development of SIFT features, SURF (Speeded Up Robust Features) features were introduced (Bay et al., 2008). They detail novel methods of generating, describing and matching feature points, building on SIFT techniques. SURF feature points are faster to generate and to compare than SIFT features. More recently, Leutenegger et al. (2011) introduces BRISK (Binary Robust Invariant Scalable Keypoints) features. They present a keypoint detection, description and matching algorithm that claims to be significantly faster than SURF.

As an alternative to matching features, features can be tracked instead. The Kalman filter (Kalman, 1960) is a computer vision algorithm used for tracking objects in video footage. The algorithm follows an object, using several frames of footage to gather information about the movement. Using the collected information it predicts what position the object will have moved to by the next frame. It then checks to see if the object is in the expected position and corrects the motion trajectory if it is not. Isard and Blake (1998) introduce the condensation algorithm for tracking. It is more robust than the Kalman filtering method with improved handling of cluttered environments.

The Lucas-Kanade method (Lucas and Kanade, 1981) presents a template matching method for optical flow estimation. Optical flow is the estimation of the motion of pixels in video footage. The Lucas and Kanade (1981) algorithm was designed for matching stereo images and uses the spatial intensity gradient. This method was developed by Tomasi and Kanade (1991) to become the Kanade-Lucas-Tomasi feature tracker for feature extraction. This uses the feature tracker of Lucas-Kanade and then defines a way of measuring the strength of matches between feature windows in neighbouring frames. It looks at the intensity differences between the windows to generate the quality of the match and gives an indication of how well a feature can be tracked.

### Edge Detection

An alternative to using feature detectors on the images is to use edge detection methods to compare edge positions between two images. Prewitt (1970), Sobel and Feldman (1968) and Roberts (1963) operators use convolution with one or more kernels to analyse the image gradient and detect edges. Marr and Hildreth (1980) use the second derivative of the Gaussian function to determine possible edge points. The Canny edge detector (Canny, 1986) uses filters to detect the strength of the edge as well as the

direction.

Martin et al. (2004) take into account a combination of colour, texture and intensity information to gain more knowledge about potential edges. This additional information gives a more detailed measure of the edge strength. Arbelaez et al. (2011) introduce a method of contour detection for use with image segmentation. They build on the Berkeley edge detector and use both local and global information to analyse brightness, colour and texture cues. They claim that their contour detector is at least as or more precise than other existing techniques, for most images.

Some set shift examples have very subtle shifts in the scenery, which are subpixel movements. A subpixel shift is when the movement is smaller than one pixel recorded by the camera. This means that there will be a jitter in the captured image due to the movement but it will not be a full pixel movement. There is a significant amount of work in the area of subpixel accuracy for edge detection (VisionBib, 2014).

Some methods of subpixel edge detection use moments to locate the edges in images (Tabatabai and Mitchell, 1984; Lyvers et al., 1989). Ying-Dong et al. (2005) introduces a method of detecting sub pixel edges using the Sobel operator to find possible edge points and then using Zernike moments to locate the edge more precisely. Additionally, the authors suggest using a preprocessing step to eliminate noise from the image. Jensen and Anastassiou (1995) develop an edge-fitting operator which uses a low-resolution version of the image to find the rough edge and then a high-resolution grid to locate the edge more precisely.

Some contour detection methods are developed to aid with image segmentation, such as that of Arbelaez et al. (2011). When fixing set shift, a segmentation tool to separate between shifted and correct areas in the image would be very useful. For example, if a shifted object is partially occluded by a moving foreground object, it is important that the foreground object is not affected when the background is corrected. The recent research into rig removal involved completing an in depth investigation into segmentation for use in stop motion footage and the related literature is analysed in section 2.2.2.

Once a shift has been detected, either by matching or tracking salient features, then the shift needs to be corrected. Research into image warping and morphing is investigated as potential ways of correcting the set shift.

#### Image Warping and Morphing

If a shift has been detected between the source and target images then a way of translating the feature positions of the shifted frame into those of the correct frame is required. Image morphing has been a popular area of research for visual effects, particularly in the 1990s. Wolberg (1996) presents an overview of image morphing algorithms, describing methods such as mesh warping, which was implemented at Industrial Light and Magic. A mesh warp takes two images and their associated meshes as input and transitions between the images by mapping their coordinates. Field morphing (Beier and Neely, 1992) was designed to try and improve the user interaction with the corresponding feature points being indicated by line pairs. It provides a simpler interface but complicates the warp generation because all the line pairs must be considered before the mapping of each point is known. Both these algorithms require an experienced user who understands the challenges of morphing to produce good

results.

There has been recent research into warping images for a variety of applications. Liu et al. (2009) introduce content preserving warps, which they use for stabilizing dynamic video content. Bai et al. (2012) present a method for 'de-animating' areas of video. Users indicate areas that they require to become static and areas that they require to remain dynamic. The algorithm warps the footage to keep the areas indicated as static in place using a variation on a technique described by Liu et al. (2009). It uses graph cuts to optimally place the warped areas of the video into the still frames, with the aim of keeping the motion natural looking. This research is focussed on large-scale movement, therefore it is unlikely to be subtle enough for the requirements of set shift.

Jacobson et al. (2011) introduce a method from which deformations can be made using a variety of weighted handles. The handles can be points, bones or cages. They have called their weights bounded biharmonic weights and they demonstrate smoother and more intuitive deformation using the range of handle types. The weights affect local areas which makes deformations react more intuitively.

Rüegg et al. (2013) present a method of compositing videos. They generate the best spatio-temporal seams to join the videos using the 3D video volume. To do this they use 3D graph cuts, which they extend to improve the handling of camera motion and efficient content matching. The alignment step in their algorithm focusses on firstly matching the correspondences between video frames and, secondly, warping so that the videos become aligned. As they do not need to align the entire frame, they suggest that homography warps work well for regional alignment.

Since this project, new research has been completed in the areas investigated. Alcantarilla et al. (2012) present KAZE features which they claim perform better than SURF or SIFT features, although are more expensive to compute. A new, robust method of feature matching is presented by Chen et al. (2013). Their method uses Hough and inverted Hough transforms to find feature correspondences between two images. Park et al. (2013) present a method of object detection that is specifically designed for video content. They use temporal difference features to provide camera and object-centric stabilisation.

Dollár and Zitnick (2013) introduce an edge detection algorithm that uses learning decision trees. It improves on the speed of previous methods while retaining the quality of results. A new contour detection algorithm is presented by Widynski and Mignotte (2012), which tracks small sections of edges. The algorithm performs well when compared to other state of the art contour detection methods.

Zhou et al. (2013) combine the strengths of homography warps with content-preserving warping (Liu et al., 2009). They claim to prevent the visible distortion which can occur in areas without texture when using content-preserving warps, while Chang et al. (2014) combine projective and similarity warps to provide good alignment for image stitching.

## 3.3 Methodology and Experiments

A prototype application was created to test some methods of detecting and correcting set shifts. The application was required to take inputs of one frame which containing shifted objects and one frame

containing the correct image content, as well as a mask indicating areas that must remain unchanged. The prototype application would be tested with single frames and if successful, it could be extended to work with frame ranges.

The proposed process of the prototype application is shown in figure 3-2. The key stages are:

1. Input: The shifted and correct images containing the set shift motion and other foreground motion which must not be affected by corrective warping. The user masks areas which are to be excluded from any corrective warping.

2. Matching: Use of a feature detection and matching algorithms to detect and match salient features between the two input images.

3. Analysis of matches: Weaker matches are removed to leave only high quality matches for use in the corrective warping. Direction of matching feature points is analysed to remove inconsistent matches.

4. Warping: The corrective warp moves the shifted features to the corrected features, based on the best matches. Surrounding pixels are affected by the warp depending on their proximity to the feature points.

5. Error correction: After warping there is likely to be a need for error correction techniques to minimize interpolation errors.

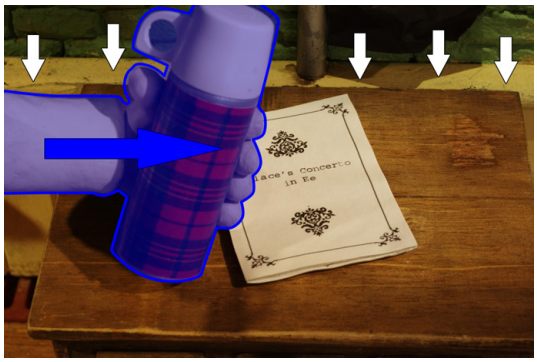
### 3.3.1 Development of the Prototype Application

The first task in the application was to find matching points between the shifted image and the correct image. This was done using an implementation of SIFT feature detection and matching (Lowe, 1999). The feature detection was run on both the shifted image and the correct image, with the most salient feature points output. The feature matching algorithm was then run using the feature point results for these two images.

The matched feature points were used to create a Delaunay triangulation between the points. Delaunay triangulation creates the optimum triangles between points by maximising the regularity of triangles and avoiding thin triangles. Using this triangulation the feature points in the shifted frame were warped to feature points in the correct frame. The barycentric coordinate system, which uses each corner of the triangle to locate the pixel position, was used to ensure that the pixels in the triangles were warped evenly.

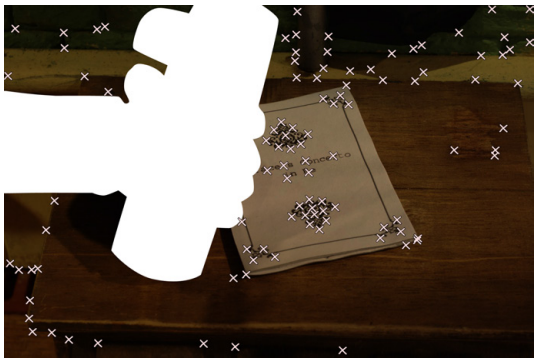
After the initial implementation of the prototype tool, it was clear that there were challenges at stage 3 of the process. Outputting coherent features between the two images was very difficult due to the subtlety of the shifts. Therefore, several different methods of detection and matching were investigated.

Difficulties with using feature point detection methods meant that contour and edge detection algorithms were also tested in this prototype application. One implementation was a contour detection method (Arbelaez et al., 2011) (section 3.3.3) and the second implementation uses Gaussian kernels (section 3.3.4). These methods also posed challenges, which led to the exploration of optical flow methods as a way of detecting shifts accurately. Optical flow is the estimation of the motion of pixels in

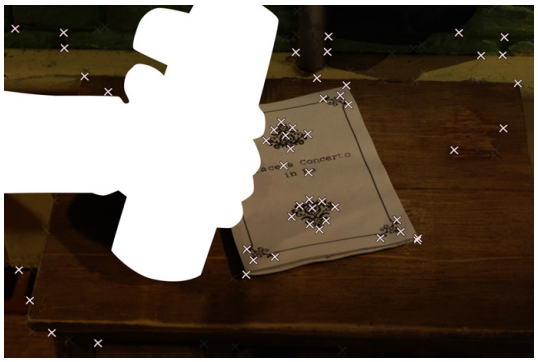


(a) Input 1: The original image, containing a subtle set shift downward in the background (indicated by white arrows). Animated movement must be retained (blue arrow) by the user masking areas to be unaffected by the warp (blue mask).

(b) Input 2: The clean plate showing the correct positioning of the background set.



(c) Feature detection and matching finds corresponding salient features (white crosses) between the unmasked area of Input 1 and Input 2.



(d) Low quality and/or inconsistent matches are discarded so that only the highest quality matches are used in the corrective warp.

(e) A corrective warp is performed based on the matching features. The warp is based on features found on Input 1 (white crosses) being moved to the corresponding feature point on Input 2 (red crosses). Surrounding pixels are warped if affected by the movement of the feature point.

Figure 3-2: The stages of the algorithm initially proposed for a tool to fix set shift ©Aardman Animations Ltd 2014.

video footage. Two existing optical flow implementations were investigated: the optical flow information available from Nuke's Vector Generator plug-in (The Foundry, 2013a) (section 3.3.5) and the optical flow information available using PFTrack (The Pixel Farm, 2014) (section 3.3.6).

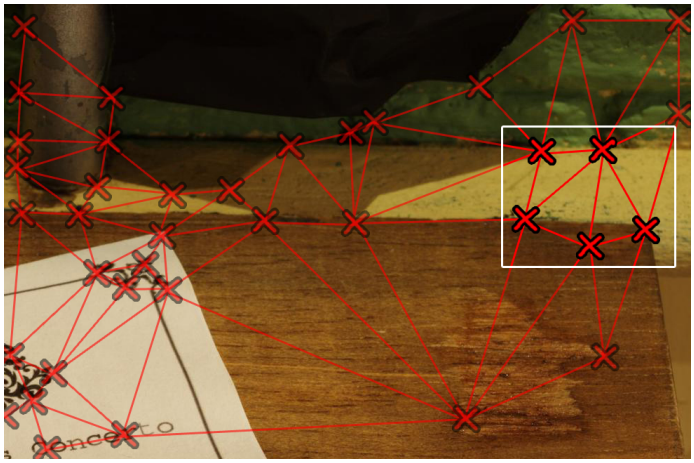


Figure 3-3: The outlined area demonstrates an example problem area when using the Delaunay triangulation for warping. As the triangles cross edge boundaries, bends can appear in the edges after warping.

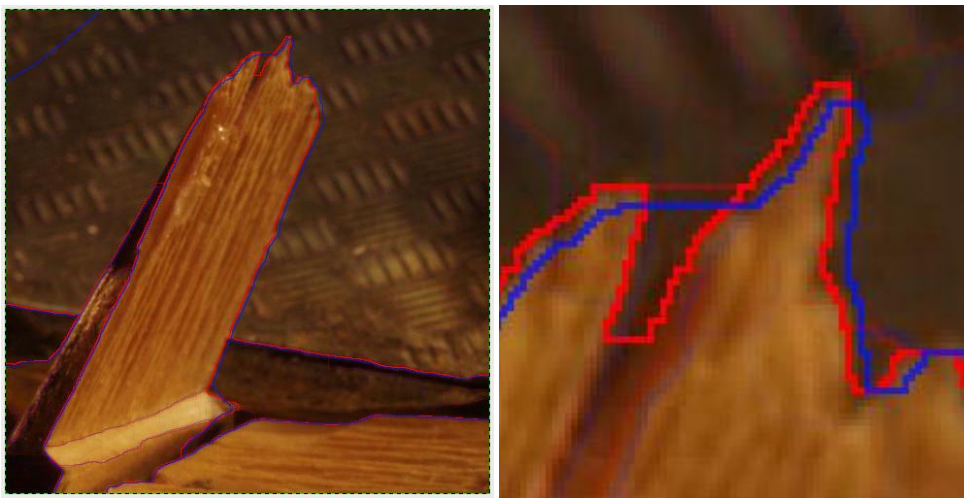
### 3.3.2 Feature Detection

The SIFT feature detection was able to locate an exaggerated set shift. However, set shifts in stop-motion production were often more subtle than the SIFT feature matching algorithm could detect. This meant that in many cases warping did not happen because the feature points were detected to be in the same position. It became clear that this prototype needed a more accurate method of feature detection and matching. Additionally where edges crossed triangles in the Delaunay triangulation, the corrective warp could leave visible bends in what should have been straight edges. Figure 3-3 shows an example of when a resulting triangulation crosses edge boundaries. To try to protect edges from bending, research was conducted into detecting the edges in a frame.

### 3.3.3 Detecting Edges using Contour Detection

Using SIFT feature detection resulted in problems with accuracy and the difficulty of erroneous warps across edges. Edge detection methods were researched in place of the feature detection component of the prototype application.

An implementation of a state of the art method of contour detection (Arbelaez et al., 2011) was tested. However there were inaccuracies when using this method on production footage. An example problem area is shown in figure 3-4, where there is a lack of coherence between the shifted and correct images. With these inconsistent edges a corrective warp could not work accurately and areas would get stretched where the edges do not align correctly. Figure 3-5 demonstrates the challenges of subpixel set shifts. The algorithm of Arbelaez et al. (2011) does not detect the subpixel movements in the set shift.



(a) (b)

Figure 3-4: Contour detection using the algorithm of Arbelaez et al. (2011). The red lines represent the edges detected in the shifted frame and blue lines represent the edges detected in the correct frame. (a) shows that overall the coherence between shifted and correct frames is good, however (b) demonstrates that there are inconsistencies in the detail, which would cause errors when warping.

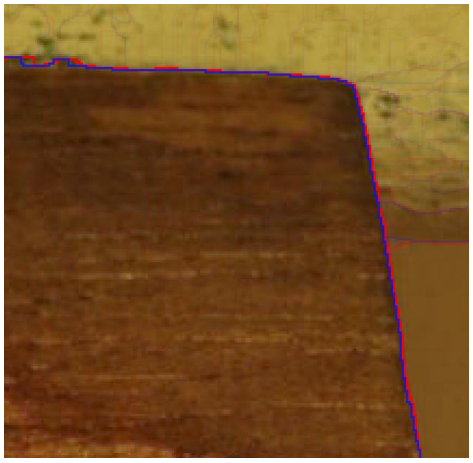
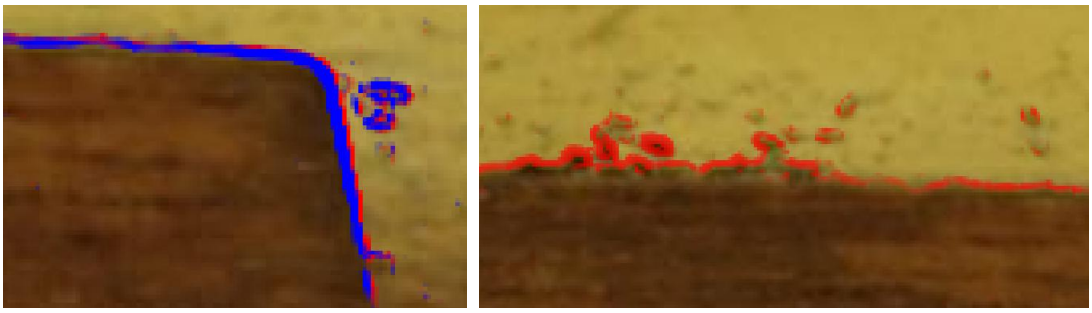


Figure 3-5: Contour detection using the algorithm of Arbelaez et al. (2011). Red lines represent the edges detected in the shifted frame and the blue lines represent the edges detected in the correct frame. This shift moves up and right but only the left-to-right movement is captured due to subpixel movement in the up-down direction.

### 3.3.4 Detecting Edges using Gaussian Kernels

Due to the challenges associated with using the algorithm of Arbelaez et al. (2011), an edge detection method using Gaussian kernels at varying scales in the Fourier domain was tested. This method had the advantage of working much faster on production images and did manage to find more coherent results between the two images (figure 3-6a). However, it struggled to pick up some of the small set shift movements. Additionally, on some images it was not accurate enough to find the correct edges due to artefacts present in the image (figure 3-6b).



(a) (b)

Figure 3-6: Edge detection using a the Gaussian kernel algorithm. Red lines represent the edges detected in the shifted frame and the blue lines represent edges detected in the correct frame (a) demonstrates issue of subpixel shifts going undetected and (b) demonstrates the difficulty of getting an accurate edge in noisy data.

### 3.3.5 Optical Flow Information using Furnace Vector Generator

Methods of generating optical flow information were researched in order to help identify the boundaries between shifting regions. Figure 3-7 depicts optical flow information using Furnace's Vector Generator plug-in for Nuke (The Foundry, 2013a). The optical flow data is set to be at maximum density, with optical flow information generated for every pixel. Figure 3-7 shows that a lot of motion information is output and it is not be possible to use this information to ascertain where the set shift occurred. There is a lot of other motion in the frame which is unrelated to the shift, perhaps due to lighting and shadow movements.

### 3.3.6 Optical Flow Information using PFTrack

The optical flow generation of PFTrack (The Pixel Farm, 2014) was also tested. Figure 3-8 shows the optical flow information output by this tool. It clearly shows the flow of the moving foreground object. Figure 3-9c demonstrates some shifting movement in the background. However, it is not consistent across the area, which could lead to uneven warping. Figures 3-9a and 3-9b illustrate additional challenges where movements are detected even when there is none visible to the eye. This could be from

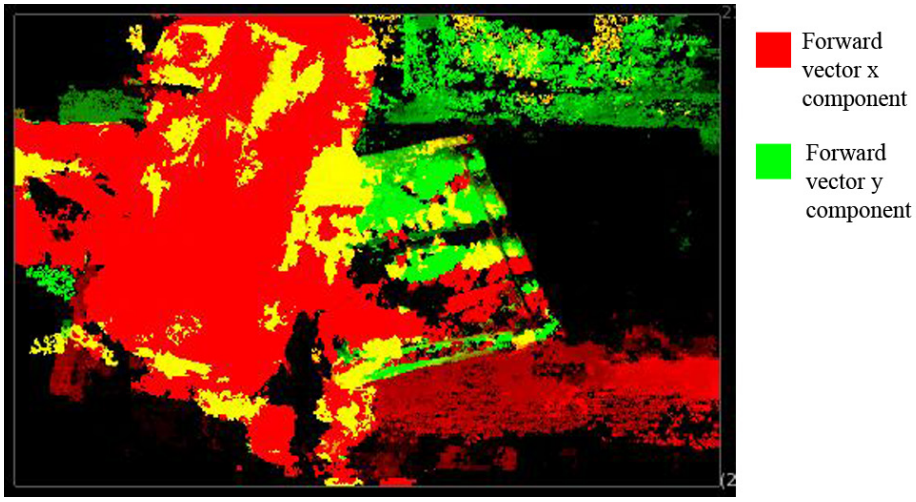



Figure 3-7: The results of generating optical flow data using Furnace Vector Generator (The Foundry, 2013a) for Nuke.

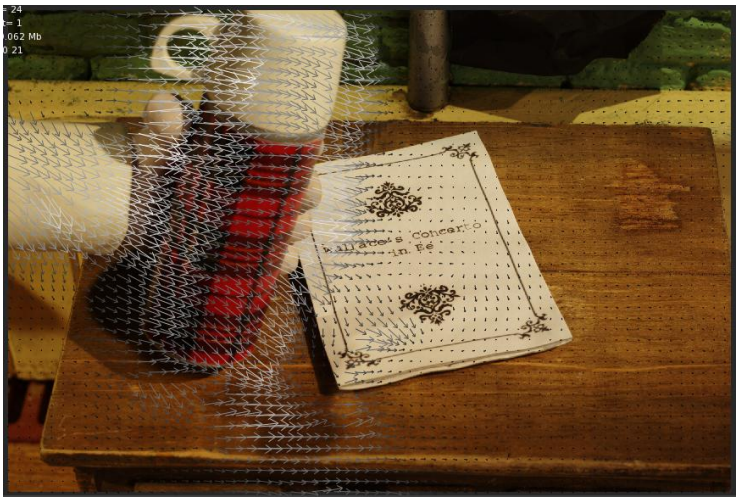
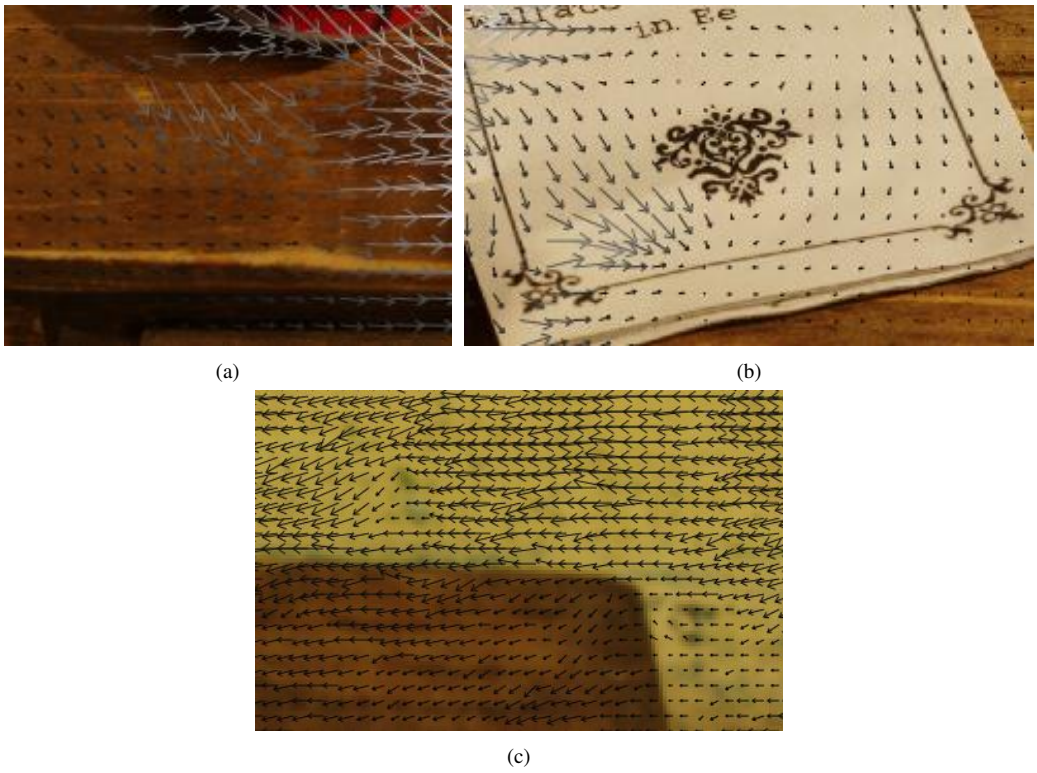


Figure 3-8: The results of generating optical flow data using PFTrack ©Aardman Animations Ltd 2014.

shadows and lighting changes from the foreground movement. These errors make accurate detection of shifted areas using optical flow very difficult.

### 3.3.7 Results

A summary of the results found during the research into feature detection is shown in table 3.1. In addition to the identified complexities of shift detection, further investigation would be needed into challenges associated with the subsequent stages of the set shift process outlined in the initial prototype. For example, challenges associated with artefacts left by the warping algorithm or with the correction of interpolation errors could require in depth investigation and further research.



(a) (b)

(c)

Figure 3-9: Details of optical flow data generated using PF Track. (a) and (b) demonstrate inconsistencies in the optical flow data caused by lighting and shadow movements on set. (c) shows the optical flow data from an area in which there has been a set shift ©Aardman Animations Ltd 2014.

| **Method** | **Challenges** |
|---|---|
| Feature detection | Incorrectly warped edges between features<br>Undetected subpixel shifts |
| Edge detection: Contour detection | Inconsistent edge results between source and target<br>Undetected subpixel shifts |
| Edge detection: Gaussian kernel | Inaccurate edge detection<br>Undetected subpixel shifts |
| Optical Flow: Furnace VectorGenerator | Complexity of output motion flow information<br>Movement detected on stationary objects |
| Optical Flow: PFTrack | Inconsistent movements between pixels<br>Movement detected on stationary objects |

Table 3.1: Summary of the methods investigated and the challenges found, for detecting set shifts in stop-motion animation frames.

## 3.4 Discussion

The results using SIFT feature point detection (Lowe, 1999) uncovered some significant challenges. Firstly, the detected points were not accurate enough when tested on stop-motion production frames. The method was more reliable when tested on exaggerated shifts but typical set shifts on production are very small movements and would require a subpixel feature matching algorithm. Secondly, there was the issue of edges crossing multiple triangles after the triangulation stage. This meant that, when the corrective warp was applied, edges could bend where they should remain straight. If these bends did happen in the warping stage then they would require complicated repair work. Due to these problems using feature point detection, it was decided to look further into edge detection to try to eliminate this problem of warping edges.

Figures 3-4 and 3-5 show results using a contour detection algorithm by Arbelaez et al. (2011). When analysing the results, it was observed that the edges are not always consistent between shifted and correct images. As the next stage of the process is to warp the edges of the source image to match the target, the results of the final corrective warp would be unreliable, resulting in visible artefacts in the final image. In addition to the differences in the edges, figure 3-5 demonstrates another problem. The shift shown moves up and right, whereas the results of the edge detection algorithm only show the shift to the right has been detected. It is thought that this is due to it being a subpixel shift. Using this edge information for a corrective warp would only lessen the set shift and user intervention would still be required to fix the shift completely.

An edge detection algorithm which uses Gaussian kernels in the Fourier domain was tested. Results showed that although this algorithm detected some edges very well, other edges were not found at all. As with Arbelaez et al. (2011), only the left-to-right shift was detected and not the up-down shift (3-6a). Additionally there were problems with detecting edges accurately (3-6b). These erroneous edges would result in incorrect warping if used in the final set shift tool. Furthermore, this difficulty in detecting edges correctly is only enhanced when shifted areas are out-of-focus, which is common as shifts often occur in the background.

It has been concluded that the level of accuracy available from state of the art contour and edge detection is not robust enough for a production tool. A tool developed using these methods would leave some areas warping correctly and others warping incorrectly. This would leave the user with an inconsistently adjusted image which would be more complicated to correct than the original image. Due to the challenges associated with edge detection, optical flow was explored as an alternative way of detecting these shifts accurately.

Two existing optical flow solutions were analysed and the results show that neither of them are accurate enough to be used in a solution for set shift. An improved optical flow algorithm would need to be developed in order for this tool to reach the standard required for stop-motion production. It would need to improve accuracy significantly in comparison to those methods tested. It would also need to output dense optical flow data, as using sparse optical flow information would make it difficult to know the location of the boundaries between movements.

At the time of research it was not possible to develop an automatic tool that could solve the problem of set shift. Since completing work on this project, new research has been published which could

help with the development of such a tool. It would be interesting to use KAZE features (Alcantarilla et al., 2012) and to test if they are more accurate than the SIFT features used in this implementation. However, sparse feature detection does not have any way of fixing problems such as bending edges in corrective warps. For this reason it would be interesting to implement the contour detection algorithm of Widynski and Mignotte (2012) as their method involves the tracking of sections of edges. If this was accurate enough then the information generated could be used to maintain edges during a corrective warp. Finally, research into video stabilisation, such as that of Park et al. (2013), could contribute to the development of an automatic solution to set shift, due to the similarly subtle shifts occurring in video footage.

## 3.5 Conclusions

In-depth investigation into possible solutions for a tool to correct set shifts has been conducted. I have concluded that current research in computer vision, in particular the areas of feature, contour and edge detection, is not sophisticated enough to produce the level of results required by a production tool. The time and resources it would take to research and develop a tool to this level is beyond the scope of this project.

Aardman Animations requested to know whether it was possible to develop a technical tool to fix the problem of set shift within the time frame available. After developing a prototype application and researching the associated challenges, it is felt that it would not be possible to develop a tool that could produce the quality required. The company had stated in the initial stages that a tool would need to work extremely well for it to be used and that if time-consuming tweaking was needed then artists would continue to use current methods. Therefore, it was concluded that this project could not be completed to the level required within the time period available. From looking at a range of the set shift examples, it is clear that there can be very subtle boundaries between shifted and correct areas, which is something that currently, only human intervention can determine. Production artists complete the repairs based on their own perception and currently it would be very difficult to reproduce this accuracy computationally.

During the investigation, a number of more complex set shift examples were found. Some set shift examples would require the development of additional functionality to those methods tested, to address the following:

- Subpixel shifts: set shifts were too subtle for feature detection and edge detection techniques.
- No visible edges: if none of the edges of one surface are visible, then a different form of analysis, such as texture analysis, would be needed to know where the surface had moved.
- Occluding edges: in some shifts one would need to know which were the occluding edges in an image. For example, if one object moves up and another moves down behind it then the algorithm could not be sure which section should occlude the other.

It was learnt from the previous rig removal project that the quality of the finish was critical. Therefore, this research began by ensuring that the problem and the required output was understood in detail.

Discussing the requirements with post-production artists gave a clear idea of what was needed from this tool. The quality of result was very important and there would certainly have been work required in the subsequent stages of the algorithm in addition to the areas requiring research discussed in this chapter. Further research would have had to be developed into the areas of corrective warping and accurate interpolation of pixels after warping. This would have needed to be very accurate so that no artefacts were left in warped areas of the footage. Additionally, once a successful method was found for individual frames, it would then need to be extended to work temporally consistently over video footage. Therefore, producing a finished set shift tool would require a significant amount of further research work, which was not feasible in the time available.

To develop a tool that would help with the problem of set shift, there are several areas in which further research could be pursued. It would be necessary to develop a more advanced understanding of the image in order to correct the shift to the required accuracy. That might require developing a new dense optical flow algorithm to provide the sub-pixel accuracy needed for detecting the shifting movements. Alternatively, a new edge detection or segmentation method could be developed to provide the accuracy required for distinguishing between the areas that need to be corrected and those which must be unaffected by corrective warping. As research in these areas of computer vision progresses it may become possible to implement production tools which can help with the problem of set shift.

The conclusion of this investigation is that the challenges associated with building a tool to the required standard are too involved given the time and resources available. This research provides Aardman Animations with the knowledge of where improvements in state of the art research are needed, before advancing with a set shift tool and investing development time on it. This project highlights the benefits of conducting an early stage project feasibility study. Particularly when working on industry-university collaborations, as there is a need to deliver both useful commercial tools and academic novelty in a relatively short timescale.

As this project has been deemed not feasible for further investigation, research projects in other areas of stop-motion animation production are to be investigated. It has been decided that the next project to research is to investigate plasticine materials, which are often used when creating stop-motion footage. The aim of understanding the material properties of plasticine being to develop a bespoke surface shader, which could accurately define the way that computer generated plasticine interacts with lighting.

# Chapter 4

# Plasticine Shading

## 4.1 Introduction

Plasticine is commonly used for creating characters and objects in stop-motion animation production. Aardman Animations are famous for their 'claymation' style of animation and their stop-motion plasticine-based animation technique is an important part of their portfolio (figure 4-1).

In addition to generating plasticine content on set, there is a need for plasticine materials to be simulated using computational methods. For example, when the production of some stop-motion content has finished but an additional element is required in post-production. A plasticine model might be created computationally and composited into the original stop-motion shot. Alternatively, some clients choose for their content to be computer generated but will request that certain elements are made to look like they are plasticine to incorporate an element of traditional stop-motion style.

Figure 4-1: Examples of plasticine being used in the creation of stop-motion footage ©Aardman Animations Ltd 2014.

In these scenarios, plasticine materials have to be recreated using surface shaders and texture maps which are applied to a computer generated model. Currently when shading plasticine materials in production, a standard surface shading material based on the Blinn (1977) model is used. In this document this material will be referred to as the benchmark model and I will investigate whether it is a good approximation of plasticine materials.

The appearance of a material depends on the way it reflects light and these reflective properties can be represented using a Bidirectional Reflectance Distribution Function (BRDF). No prior research exists into defining a BRDF for plasticine materials. However, there is a lot of research into surface shading (Section 4.2), including studies of specific materials, such as hair, cloth and skin.

In this research, a new, more accurate surface shading model for plasticine materials is proposed, which is based on the true physical properties of plasticine. To understand the surface properties of plasticine in detail, BRDF acquisition is used (section 4.3.1). The final model is based on microfacet theory and builds on the shading models of Walter et al. (2007) and Cook and Torrance (1982). The proposed model is tested along with a number of state of the art models, by fitting the results to the recorded BRDF data (section 4.3.4).

The research presented in this chapter delivers the following contributions:

- The Plasticine model: a physically-based shader combining elements of two existing microfacet shaders. This model simulates the reaction of plasticine to light more accurately than using existing models independently (section 4.4.1).
- The PlasticinePlus model: extends the Plasticine model to enhance the behaviour, in particular when the incident light is at grazing angles (section 4.4.2).
- Recorded BRDF data for plasticine materials, using the parameterisation of Rusinkiewicz (1998) and the same format as the MERL materials database (MERL, 2006) (section 4.3.1).

The results (section 4.5.1) show that the new model simulates plasticine more accurately than other models, with a reduction in the fitting error of around 20% when compared to the next closest state of the art model. The improved plasticine model has been implemented into production systems and the new model is compared to the benchmark model. The results demonstrate an improvement in the realism of simulated plasticine, which is attributable to the work completed into understanding and interpreting the physical properties of plasticine.

## 4.2 Background

### 4.2.1 Production Background

Currently, when a computer generated model needs to be shaded to look like it is made from plasticine, an off-the-shelf shader is used to simulate the material. Typically, a shader based on the model of Blinn (1977) is used, with the parameters adjusted to make the reflectance properties similar to those of plasticine. Additionally, texture is added to the surface using a displacement map. The displacement map is created using images of fingerprints and other marks, to give the appearance of plasticine and to simulate the indentations that are visible on plasticine models. A separate colour or diffuse map is also added to simulate the variation and flecks of colour across the material. The shading, colour and the texture models must be reproduced accurately in order for a realistic simulated material to be generated. This research focusses predominantly on the generation of a realistic surface shading model.

**Limitations**

As this project is being completed for Aardman Animations there are some further requirements which will influence the direction of the research.

- They have specified that they would like the plasticine material to work with the V-Ray renderer (Chaos Group, 2014). It must be written as a plug-in for V-Ray and must interface with Maya.
- They require the model to be physically plausible and that it will retain some of the physical properties of real plasticine. This will help to ensure that the shader is a convincing plasticine simulation.
- The shader must be user-friendly. The parameters provided should be intuitive to the user and allow a range of plasticine looks to be achieved. This objective may conflict with the previous limitation, in which case the limitations will need to be balanced.
- For the purposes of this research I have assumed that perfectly smooth plasticine is an isotropic material.

### 4.2.2 Related Literature

A BRDF defines the proportion of light reflected from a material, given the angle of the incident light ($I$), the angle of the reflected light ($O$) and the surface normal ($N$) (Dorsey et al., 2008). It represents the ratio of radiance to irradiance at a given point (equation 4.1).

$$BRDF(\theta_I, \phi_I, \theta_O, \phi_O) = \frac{L_O}{E_I} \tag{4.1}$$

where $L_O$ is the outgoing light or surface radiance and $E_I$ is surface irradiance.

Radiance and irradiance are both measurements of light intensities. Irradiance measures the light intensity at a point, often the light intensity incoming to that point (figure 4-2), whereas radiance measures the light intensity in a particular direction, often the amount of light reflected from a point in a certain direction. Irradiance is measured in power per area and radiance is measured in power per area per steradian. Therefore, the unit of the BRDF is an inverse steradian.

Marschner et al. (2003) state that they need to know three things in order to interpret a pixel value as a BRDF measurement. They need to know how responsive the pixel sensor is to scene radiance, what the irradiance is at each surface point and what the surface normal, lighting direction and viewing direction are at that surface point.

**Irradiance** The irradiance can be calculated by measuring the incoming light energy at a point (equation 4.2).

$$E_I = \int_{\Omega_I} L_I cos\theta_I d\omega_I \tag{4.2}$$

where $L_I$ is the incoming light, $d\omega_I$ is the differential solid angle of the incoming light and $cos\theta_I = N \cdot L$ .

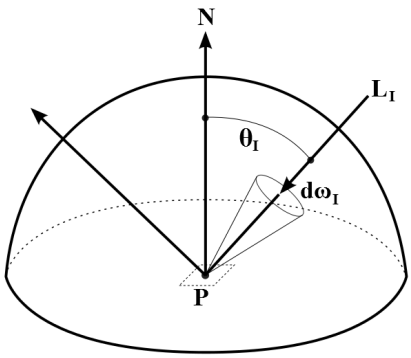



Figure 4-2: Irradiance describes the incoming light energy at a given point.

**Radiance** When measuring BRDF data, the mapping of scene radiance to pixel radiance is recorded in a camera's response curves. Each model of camera has a different set of response curves mapping the recorded pixel value to the real scene radiance at that point. CCD sensor cameras are popular for recording BRDF data as they have a close-to linear response curve, particularly when used in a chilled environment. CMOS sensor cameras can be used but the response curves must be known or calculated, so that recorded data can be converted to be directly proportional to the scene radiance. Response curves can be generated for each of the RGB colour channels, by taking sample images at a range of exposures. This can be done using external software (Jaloxa, 2011) or manually (Robertson et al., 2003). Once the response curve of the camera is known, the camera data can be processed using the functions associated with the curves. This calculated value will be directly proportional to the scene radiance (Narasimhan, 2008).

There are many analytical models which approximate BRDFs and the relationship between irradiance and radiance mathematically.

### 4.2.3 Analytical Models

Some of the earliest and most well known analytical shading models are the Blinn (1977) and Phong (1975) models. More recently, physically-based or physically plausible shaders have become increasingly popular (Cook and Torrance, 1982; He et al., 1991; Ward, 1992; Oren and Nayar, 1994; Ashikhmin and Shirley, 2000). These shading models are developed to obey the basic laws of physics. In general it is expected that they will follow two laws; energy conservation and reciprocity. Energy conservation states that the total incoming energy equals total outgoing energy plus any absorbed energy, where BRDF models tend to assume that the energy absorbed is negligible. Reciprocity means that the BRDF is symmetric with respect to incoming and outgoing directions.

The balance between physical realism and artistic flexibility is something that is important in the development of a plasticine shader. Disney have adopted physically-based shading techniques for many of their materials (Burley, 2012). For example, when building a hair model for the film "Tangled" they completed extensive research into physically-based hair shading. They then refined their physical

model to make it artist-friendly (Sadeghi et al., 2010). When Disney developed their material shading system, they began by investigating the physical attributes of materials but developed their findings to incorporate artistic metrics (Burley, 2012). Their final shaders use a physically principled model rather than a strictly physical one.

A common model on which physically-based shaders are developed is microfacet theory (Hoffman, 2012). This theory dictates that each surface consists of many small, unseen microfacets which contribute to the appearance of the surface. Three elements contribute to the final appearance of the surface: The microfacet distribution model (D term), the geometry, or shadow-masking, function (G term) and the Fresnel value (F term). Each are modelled mathematically.

- D term: The microfacet normal distribution model describes the distribution of microfacets over the surface. D simulates the number of microfacets which are facing towards, and will therefore reflect, the light.

- G term: The geometric attenuation factor accounts for the fact that some microfacets will be shadowed or masked and therefore will not reflect any light. G represents the percentage of the microfacets which are not shadowed or masked. This is very important as it ensures that the BRDF cannot reflect more light energy than it receives and thus satisfies the law of energy conservation.

- F term: The Fresnel component represents the reflection of the microfacets and how much light they would reflect if optimally flat. The index of refraction of the material is needed in order to calculate the amount of light reflected given the light direction and the surface normal. The Fresnel term predicts that smooth surfaces will approach 100% specular reflection. Rough surfaces will not reach this level of reflection but the surface will still become increasingly specular (Burley, 2012). The Fresnel term is not always used in shaders as it is the most time-consuming to compute. It can be modelled fully, or by the Schlick (1994) approximation which is faster and can provide comparable results.

These D, G and F terms are combined to model the specular function of the material, once divided by a correction term which varies depending on the models used. Typically this specular component is added to a diffuse component to create the final surface material. Many different models use microfacet theory to simulate materials, for example, the Blinn (1977) model can be split into D, G and F components.

A drawback of the microfacet model is that it ignores inter-reflections between the microfacets and assumes that the light does not bounce around in the shadows between microfacets. Additionally, the microfacet model does not account for pronounced wave optic effects such as diffraction or interference (Hoffman, 2012).

Cook and Torrance (1982) and Torrance and Sparrow (1992) are examples of popular microfacet models used for representing materials and fitting BRDF data. The D, G and F terms of microfacet models can be researched individually, such as the model of Smith (1967) which represents geometrical self shadowing properties. Sometimes terms may be tailored to suit a specific material, for example Walter et al. (2007) introduce a new microfacet distribution model called GGX which they find fits rougher surfaces, such as ground glass, more accurately.

In order to develop a physically accurate analytical model to represent plasticine, information about the surface properties of plasticine material must be investigated. No prior research exists on the acquisition of data from plasticine. However, there are many other specific materials which have been investigated in order to generate realistic and physically plausible shaders. Examples of such studies include, but are not limited to, car paint (Günther et al., 2005; Rump et al., 2008), cloth (Wang et al., 2008; Sadeghi et al., 2013), skin (Donner et al., 2008; Marschner et al., 1999) and hair (Marschner et al., 2003). As a production example, BRDF acquisition was used for simulating the red suedette material of Superman's cape in the film "Man of Steel" (Schwarz, 2012).

### 4.2.4 Acquisition of BRDF Data

There are multiple ways of acquiring BRDF data from sample materials. In order to populate a table of BRDF data, it is necessary to capture how a material reacts to light in all combinations of light direction, view direction and surface normal. This is commonly known as BRDF acquisition. Modelling and acquiring BRDFs is an active research area. Experiments range from being relatively simple and easy to set up, to complex methods requiring specialist equipment.

Measured materials are either isotropic or anisotropic. Isotropic materials can be rotated about the surface normal and the reflections will not change, whereas the reflections on an anisotropic material will vary dependent on the direction. For example, wood is an anisotropic material as the grain affects the surface reflections differently dependent on the viewing direction. Anisotropic materials have an added degree of freedom which must be recorded in the BRDF table and more samples need to be taken when acquiring data for an anisotropic material than for an isotropic material.

Specialist equipment such as a gonioreflectometer (Foo, 1997) can be used for acquiring data for both isotropic or anisotropic materials. This device consists of a material sample, a light and a sensor to capture the amount of light reflected from the material. Both the sensor and the light can rotate around a hemisphere of the material so that many combinations of lighting and viewing direction can be captured. These devices have to be carefully calibrated to ensure they record accurate measurements for each combination of angles. In the design of Foo (1997), the light source rotates around it while the detector remains stationary and records high-definition data. Anisotropic materials can be captured using equipment such as gonioreflectometers and parabolic mirrors (Filip et al., 2013).

In this research it has been assumed that perfectly smooth plasticine is isotropic. The BRDFs of isotropic materials can be acquired using less complex experiments. Marschner et al. (2003) develop a method of recording isotropic BRDF data using a sphere of the target material, controlled lighting and a digital camera. The use of a spherical sample helps in the acquisition of BRDF data as it allows many angles to be captured in one image. Matusik et al. (2003) have captured data for a wide range of isotropic materials which are accessible in the MERL database (MERL, 2006). The methods for capturing the data and generating BRDF tables for each material are discussed. Matusik (2003) creates an automatic capture system and records a large number of materials to represent dense BRDF data. Ghosh et al. (2010) use a mirrored dome and a beam splitter to approximate the BRDF.

Günther et al. (2005) build on the work of Marschner et al. (2003) and Matusik et al. (2003) to present a method for recording BRDF data in order to develop an accurate car paint shader. Günther

et al. (2005) also use a spherical sample of their material, positioned on a turntable along with an arm holding a lamp. This turntable, and therefore the lamp and sphere, is rotated in increments whilst the camera stays static. One hemisphere of data is recorded to populate the BRDF table.

Marschner et al. (1999) develop a method to allow for capture of BRDF measurements from irregular objects. A 3D model is obtained using a scanner and reflectance data is calculated using this surface information. Their method works for isotropic materials and in particular, they look at the capture of reflectance data from human skin. Recent research by Ali et al. (2012) demonstrates a way of acquiring BRDFs from materials using less samples. Ashikhmin and Premoze (2007) present an alternative to capturing a full table of measured data. They reduce the data capture required significantly but the new model of representing BRDF data is "not strictly physics-based". They require only backscattering data and some estimation of the Fresnel value which means that a lot less data needs to be captured from the material. A drawback of this method is that these values are more difficult to measure.

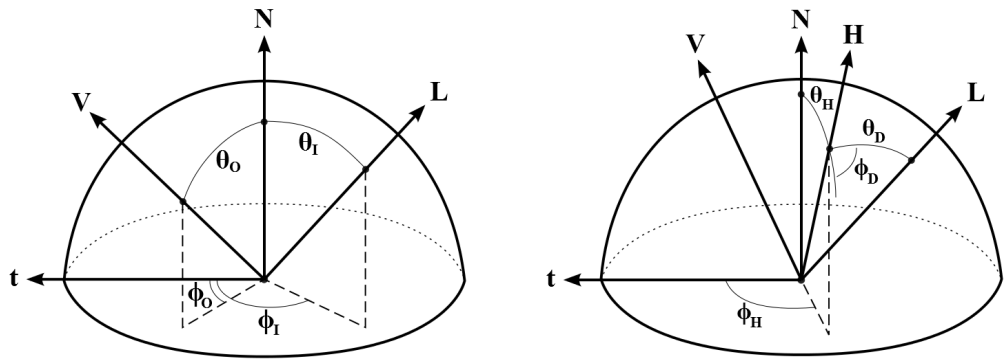
One challenge of BRDF acquisition is gathering accurate data at grazing angles or near the point of retro-reflection. Ngan et al. (2005) state that they ignore measurements at angles greater than 80° during data fitting due to the unreliable nature of the data.

### 4.2.5 Processing the Recorded Data

With a large volume of data recorded in the captured images, the images must be processed to populate a BRDF table. Matusik et al. (2003) use each pixel as a sample and calculate the radiance and irradiance at that position. It is important to ensure that the radiance recorded in the image is linearly related to the radiance in the scene and that radiance and irradiance are measured in relative units (Marschner et al., 2003).

Traditionally BRDFs were recorded using four parameters ($\theta_I$, $\phi_I$, $\theta_O$, $\phi_O$) where $I$ and $O$ represent the incoming and outgoing light respectively (figure 4-3a). Rusinkiewicz (1998) introduces a new method which uses the half vector, $H$, between the lighting direction and the viewing direction. The half vector is used in conjunction with the difference vector, $D$, to parameterise the BRDF data (figure 4-3b). Günther et al. (2005) state that using the half vector combined with non-linear sampling of $\theta_H$ allows for a more accurate representation, particularly of the specular highlight . Furthermore, for isotropic materials the value of $\phi_H$ can be discarded which reduces the parameters to three. Stark et al. (2005) present a parameterisation of BRDF data that further reduces the BRDF parameters to two values.

Günther et al. (2005) choose to represent their data using the parameterisation of Rusinkiewicz (1998). Each pixel sample is processed and the $\theta_H$, $\theta_D$ and $\phi_D$ values are calculated at that point. These parameters are used to populate the appropriate bin in the BRDF table. When there are multiple values entering one bin, Matusik et al. (2003) remove outliers by removing the top and bottom 25% of values and average the remaining values. If there are no values for a particular bin then the table is filled with a weighted sum of the neighbouring bins (Günther et al., 2005). In the BRDF acquisition outlined by Marschner et al. (2003) and Günther et al. (2005) each colour channel is treated separately. The resulting BRDF table has three sets of $\theta_H$, $\theta_D$ and $\phi_D$ values representing each of the RGB colour channels.




(a) Standard parameterisation

(b) Rusinkiewicz parameterisation

Figure 4-3: The Rusinkiewicz (1998) parameterisation in comparison to the standard parameterisation for representing BRDF data.

### 4.2.6 Data Fitting

Once the BRDF table has been acquired it is common to fit the data to an analytical model (Ngan et al., 2005; Ward, 1992; Lafortune et al., 1997). The process of fitting the data involves measuring the differences between the values in the acquired BRDF table to values generated by a mathematical or analytical shading model. The parameters of the analytical model are optimised to find those which generate the lowest difference in values, or lowest error. The best analytical models for fitting arbitrary BRDF data have been studied. Ngan et al. (2005) compare existing analytical models using the MERL dataset of 100 isotropic materials (MERL, 2006) and their own BRDF measurements for four anisotropic datasets. The results show that the models of Cook and Torrance (1982), Ashikhmin and Shirley (2000) and He et al. (1991) produce the lowest error. They suggest that a contributor to their good performance is that they each model the Fresnel effect explicitly.

Günther et al. (2005) fit their BRDF measurements to the model of Cook and Torrance (1982) using a constrained Levenberg-Marquardt method, which is a method used to solve non-linear least squares problems. They minimise an energy function containing the captured data and the parameterised Cook-Torrance model. The parameters are limited within physical constraints. During their analysis Ngan et al. (2005) use Sequential Quadratic Programming to generate the parameters for the specular data and use linear least squares to compute the diffuse and specular colour parameters separately. While Günther et al. (2005) place importance on the physical plausibility of the parameters, Ngan et al. (2005) run the data fitting unconstrained.

In this project, once the results of fitting state of the art models to the captured plasticine data have been obtained, I analyse the best fit models. From this analysis, a new model is developed based on the strengths and weaknesses discovered in the fitted mathematical models.

## 4.3 Methodology

### 4.3.1 BRDF Acquisition

To acquire reflectance information about the plasticine material, equipment was set up using a similar approach to the acquisition method of Günther et al. (2005). My acquisition set-up is shown in figure 4-4. A spherical sample of the material was placed in the centre of a turntable, with a lamp attached to the turntable using an arm. A tungsten light was used with the beam set to be as close to parallel as possible.

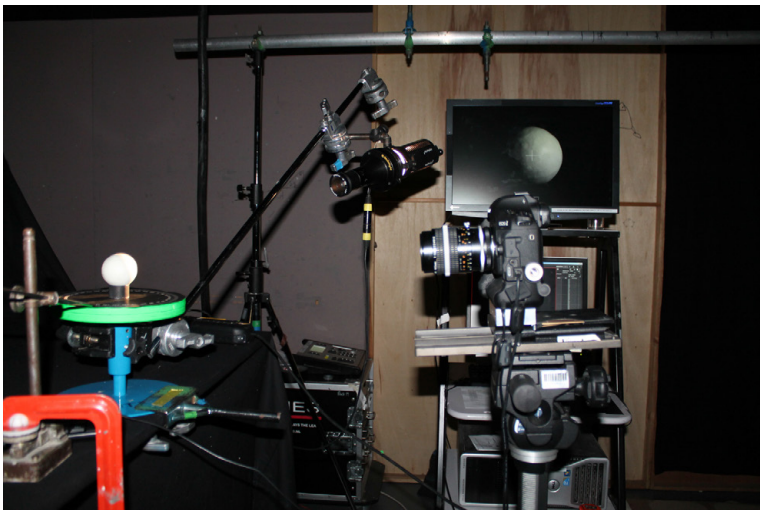


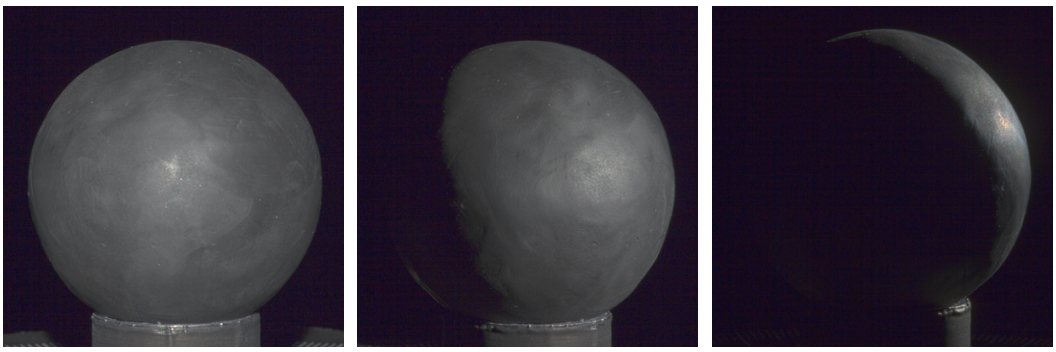
Figure 4-4: The BRDF acquisition set-up.

A stationary camera (Canon EOS 1 mark IV) was directed towards the material sample, which was positioned in the centre of the image where the camera's sensors are most accurate. A blackened room was used with the only illumination coming from the single light source. To capture a dense set of data, the turntable was rotated at 1 degree increments for 0 to 179 degrees. Images were captured at multiple exposures for each position to ensure that the full reflectance range was recorded.

The decision to use a sphere of plasticine as my sample presented a challenge. The plasticine sample needed to be as smooth as possible for both accurate geometric calculations as well as the assumption of an isotropic material. However, it is extremely difficult to create a 'perfect' sphere from plasticine. Spherical moulds were used but, due to the nature of the material, indentations were left on the surface on removal from the mould. Therefore, some texture was present on the surface during acquisition and this was visible under the experiment lighting (figure 4-5). To reduce the issues associated with having an imperfect material sample, the experiment was completed on multiple samples. For each colour of plasticine, measurements from three samples were averaged in post-processing to reduce inaccuracies created by the surface texture.

Aardman Animations use two different types of plasticine in productions, which they refer to as UK plasticine and US plasticine. The two types of plasticine have different properties and a different

look. For example the US plasticine conatins more oil and is therefore shinier, than the UK variant. The research presented here has been completed on UK plasticine, as this is the variant for which the company requested the initial shading model. For these first investigations into the properties of plasticine, it has been decided that black and white samples of UK plasticine will be used. These samples have been chosen so that any properties specific to light or dark plasticine colours will be captured. The aim is to find a shading model that will satisfy these two colours of plasticine. Following the development of a model based on these samples, other colours of the material could then be tested.



(a) (b) (c)

Figure 4-5: Images captured during the plasticine BRDF acquisition experiment.

The images in figure 4-5 demonstrate some of the properties of plasticine noticed during capture. The material looks relatively matte at forward-facing angles. However, as it approaches grazing angles, the specular highlight becomes increasingly strong. Plasticine contains oils which are visible on the surface of the material. These oils contribute to the sheen, which is particularly apparent at grazing angle illumination.

### 4.3.2 Calibration

Measurements were taken for each component in the experiment, for use when processing the data. Distances and angles between the camera, sample and light were measured along with the radius of the sample. A light meter and power meter were used to measure the light incident on the sphere, i.e. the surface irradiance.

**Capturing the Camera Response Curves** Günther et al. (2005) and Marschner et al. (2003) favoured the use of CCD cameras due to their accuracy and the linear response curve. As this type of camera was not available, I used a camera with a CMOS sensor and therefore it was necessary to calculate the camera's response curves. The response curves were calculated by taking images at a range of exposures and then uploading the images to a website (Jaloxa, 2011). Polynomial curves were generated for each of the colour channels and represent the relationship between the values stored in the camera and the values in the scene. These response curves would be used when processing the images to ascertain the linear relationship between the scene radiance and the irradiance at the sensor.

**Calibrating the Irradiance and Radiance Values** Additionally, the radiance and irradiance values needed to be captured in relative units. Therefore, an additional experiment was conducted using just the light and the camera. The camera's response to light was recorded and this data was used for conversion between pixel values and light strength. The light was pointed to look directly at the camera and the Wattage of the light was adjusted using a dimmer control. Images were captured for varying strengths of light so that the correlation between light strength and camera values could be analysed. For each image captured a power meter reading was recorded at the lens of the camera. Using these power meter readings, in conjunction with the camera data and response curves, meant that radiance values in the image and the surface irradiance could be measured in the same units.

### 4.3.3 Processing the Captured Images

After capture, the multiple exposure images were converted into high dynamic range images. A mask file was generated to indicate the areas of the image containing the plasticine sphere (figure 4-6). Every pixel in the top half of the sphere was used as a sample to generate BRDF data. This ensured that reflectance from the turntable did not interfere with the recorded data (Günther et al., 2005).

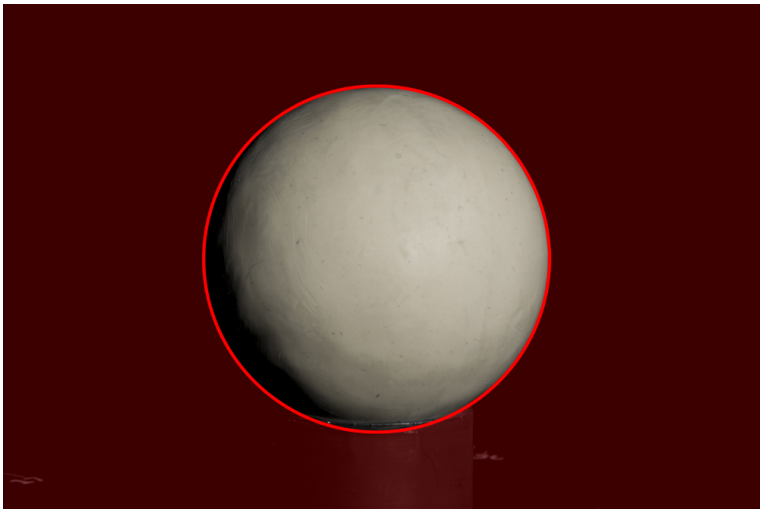


Figure 4-6: The mask applied to a captured image to control which pixels get processed.

The parameterisation of Rusinkiewicz (1998) (figure 4-3) was used for my BRDF table. Therefore, the data collected uses the same format as the MERL materials database (MERL, 2006) and can be analysed using the Disney BRDF Explorer (Disney, 2012). This application allows a BRDF table to be viewed and compared with other measured materials and analytical shading models.

For every pixel in the hemisphere the surface normal, lighting vector and viewing vector are calculated. Using this information, values are generated for $\theta_H$, $\theta_D$ and $\phi_D$, which indicates the appropriate bin in the BRDF table. This bin is populated with the ratio of radiance to irradiance at that point (equation 4.3).

$$BRDF(\theta_H, \theta_D, \phi_D) = \frac{L_O}{L_I cos\theta_I d\omega_I} \tag{4.3}$$

where $L_O$ is the outgoing light, $L_I$ is the incoming light, $d\omega_I$ is the differential solid angle of the incoming light and $cos\theta_I = N \cdot L$ .

Due to the law of reciprocity measured BRDF values can be entered into the table in two positions, i.e. the equivalent of $BRDF(\theta_I, \phi_I, \theta_O, \phi_O)$ and at $BRDF(\theta_O, \phi_O, \theta_I, \phi_I)$ when converted to the parameterisation of Rusinkiewicz (1998).

### 4.3.4 Data Fitting

An analytical model to approximate the data contained in the BRDF table was required. Mathematical data fitting was completed for several state of the art models to find the best approximation to the captured data. The aim of the data fitting was to compare the acquired BRDF values to analytical models to find which could be optimised to fit the data most closely. The best fitting model would be the one which produced the lowest fitting error. The fitting error represents the difference between the values in the acquired BRDF table and the values generated by the mathematical model for each of those table positions. In my implementation of data fitting, the fitting error represents the minimised sum of the squares of the errors.

For ease of implementation, the data fitting used a least squares approach with the constraints set to limit parameters to a physically plausible range. Non-linear least squares iterative optimisation was completed in MATLAB, using code such as:

```
[fittedCoeffs, resnorm] = lsqnonlin(@(x)minimizeCookTorrance(x,
   angleData, accumulatedData), coeffs, minBound, maxBound, options);
```

where `minimizeCookTorrance` is a function containing the shading algorithm of Cook and Torrance (1982) (equation 4.5). The data acquired from the BRDF acquisition, `accumulatedData`, is passed to the function along with the values of $\theta_H$, $\theta_D$ and $\phi_D$, `angleData`, for each entry in `accumulatedData`. The parameter values that are used to initialise the function are stored in `coeffs`. Arrays `minbound` and `maxbound` are used to limit the boundaries during the iterative optimisation of the coefficients. The function returns the optimised coefficients, `fittedCoeffs`, which represent the parameter values that simulate the acquired data most closely. The function also returns the fitting error, `resnorm`, which represents the minimised sum of the squares of the errors.

The values chosen to initialise the coefficients were based on visual estimation, by testing parameters in a renderer to find visually similar results. Additionally, the fitting was run with a number of different initialisations to ensure that the convergence of the optimum parameters was consistent. The boundaries were selected to ensure that resulting parameters remained positive and to ensure that they remained physically plausible, such as for the index of refraction. Example initial coefficient and boundary values for the least squares fitting are:

```
coeffs = [0.8; 1.5; 0.5; 0.8];
minBound = [0.5,1.3,0.1,0.1];
maxBound = [3.0,2.0,inf,inf];
```

where the coefficients are values $ks$ for the specular component, $\gamma$ for the index of refraction in the Fresnel term, $\alpha_G$ for the roughness parameter in the distribution term and $kd$ for the diffuse component.

Fitting was calculated for the models of Cook and Torrance (1982), Blinn (1977), a Torrance-Sparrow implementation of Blinn (Torrance and Sparrow, 1992), Ashikhmin and Shirley (2000) and Walter et al. (2007), on both black and white plasticine data.

**Results of the data fitting**

The GGX model of Walter et al. (2007) (equation 4.4) and the model of Cook and Torrance (1982) (equation 4.5) performed relatively well when fitted to the captured data. Having analysed the behaviour of the GGX model (Walter et al., 2007), it was clear that although it approximated some lighting angles well, there were areas where it did not exhibit the qualities shown in the captured plasticine data. For example, when the viewing direction and lighting direction are at grazing angles the GGX model does not demonstrate the same level of specular shine as the recorded plasticine data. Neither the GGX model nor the Cook-Torrance model offer the ability to simulate the reflective qualities, more matte at lower angles of incidence and high levels of specular highlights at grazing angles, exhibited in the captured plasticine data.

$$BRDF_{walter} = \frac{kd}{\pi} + \frac{ks(D_{GGX}G_{SmithGGX}F)}{4(N \cdot L)(N \cdot V)} \tag{4.4}$$

$$D_{GGX} = \frac{{\alpha_G}^2}{\pi \left(N.H^2({\alpha_G}^2 - 1) + 1\right)^2}$$

$$G_{SmithGGX} = \left(\frac{2}{1 + \sqrt{1 + {\alpha_G}^2 \left(\frac{1-(N \cdot L)^2}{(N \cdot L)^2}\right)}}\right)\left(\frac{2}{1 + \sqrt{1 + {\alpha_G}^2 \left(\frac{1-(N \cdot V)^2}{(N \cdot V)^2}\right)}}\right)$$

where $kd$ and $ks$ are the diffuse and specular components, respectively. $F$ is the Fresnel calculation and $\alpha_G$ is the roughness parameter.

$$BRDF_{cooktorrance} = \frac{kd}{\pi} + \frac{ks(D_{ct}G_{ct}F)}{\pi(N \cdot L)(N \cdot V)} \tag{4.5}$$

$$D_{ct} = k\mathrm{e}^{-\left(\frac{acos(N \cdot H)}{\alpha_G}\right)^2}$$

$$G_{ct} = min\left[1, \frac{2(N \cdot H)(N \cdot V)}{V \cdot H}, \frac{2(N \cdot H)(N \cdot L)}{V \cdot H}\right]$$

where $kd$ and $ks$ are the diffuse and specular components, respectively. $F$ is the Fresnel calculation, $\alpha_G$ is the roughness parameter and $k$ is a constant.

## 4.4 A Bespoke Plasticine Model

The results produced, using optimum parameters on state of the art models, demonstrated areas for improvement. State of the art models struggled to represent some of the particular qualities of plasticine materials, such as the strong specular peak at grazing angles representing the oily sheen on the surface. Firstly, I introduce a model which combines two state of the art models (section 4.4.1) and demonstrates

an improved fit to captured plasticine data. Secondly, I extend the combined plasticine model to further improve the simulation at grazing angles (section 4.4.2).

### 4.4.1 The Plasticine Model

The Plasticine model combines elements of the model of Cook and Torrance (1982) and the GGX model (Walter et al., 2007). It achieves a lower error when fitted to the captured plasticine data (table 4.1) by widening the specular peaks at grazing angles and representing the shadows more accurately in front-facing lighting. The model uses the G term from Cook and Torrance (1982) and the D term of the GGX model (Walter et al., 2007). Equation 4.6 shows the improved Plasticine shading model.

$$BRDF_{plasticine} = kd + \frac{ks(D_{plasticine}G_{plasticine}F)}{N \cdot L} \tag{4.6}$$

where $kd$ and $ks$ are the diffuse and specular components, respectively.

$$D_{plasticine} = \frac{{\alpha_G}^2}{\pi \left( (N \cdot H)^2 ({\alpha_G}^2 - 1) + 1 \right)^2} \tag{4.7}$$

where $\alpha_G$ is the roughness parameter.

$$G_{plasticine} = min \left[ 1, \frac{2(N \cdot H)(N \cdot V)}{(V \cdot H)}, \frac{2(N \cdot H)(N \cdot L)}{(V \cdot H)} \right] \tag{4.8}$$

$$F = 0.5 \frac{(g-c)^2}{(g+c)^2} \left( 1 + \frac{((c(g+c)-1)^2}{((c(g-c)+1)^2} \right) \tag{4.9}$$

where $c = V \cdot H$, $g = \sqrt{\gamma^2 + c^2 - 1}$ and $\gamma$ is the index of refraction for the material.

Using this combination of the two models ensures that the model is physically-based. Walter et al. (2007) state that "Cook and Torrance used a G based on a 1D model of parallel grooves that guarantees energy conservation for any distribution D". Therefore, the Plasticine model should still ensure energy conservation.

### 4.4.2 The PlasticinePlus Model

The Plasticine model presents an improved physical model for representing plasticine materials, which fits the recorded data mathematically more accurately (table 4.1). This model was extended by refining the modelling of the specular highlights at grazing angles.

It has been documented that as data approaches grazing angles it becomes difficult to assess its reliability. Ngan et al. (2005) suggest discounting measurements of incident angles greater than $80^\circ$ from data fitting as results at these angles are unreliable. I found that even values approaching $80^\circ$ in the captured data became less and less consistent. Therefore, when trying to simulate the behaviour at grazing angles, the visual quality of results was assessed.

Aardman Animations required that the research was based on the physical properties of plasticine. However, of primary importance was that the final model looked visually similar to the captured plas-

ticine data, in the same way that Disney's shaders were chosen to be 'physically principled' so that they allowed for the best visual results and for artistic freedom (Burley, 2012).

In order to improve the simulation at grazing angles, amendments were made to the G component of the microfacet model.

$$G_{plasticinePlus} = \left(G_{plasticine}(1 + max(0.5 - \lambda(V \cdot H), 0))^2\right)^{\beta(1-N \cdot L)} \tag{4.10}$$

The PlasticinePlus model (equation 4.10) multiplies the $G_{plasticine}$ term to enable values greater than 1 as $V \cdot H$ approaches 0. This multiplication, along with the exponent, increases the intensity of the specular component at grazing angles. Parameters allow control over the grazing shine, where $\lambda$ (typically between 0.5 and 1.5) controls the range of angles affected by grazing shine and $\beta$ (typically between 1 and 2) controls the specular strength.

### 4.4.3 Implementation Details

The Plasticine and PlasticinePlus models are implemented into a new V-Ray shading material and Maya wrapper, which has been integrated into the production pipeline. The shader interface makes it possible to switch between the Plasticine model and the PlasticinePlus model. The user can select a control to 'Force Energy Conservation' which allows those users who require a physically-based model to use the Plasticine model without the additional modifications of the PlasticinePlus model.

The parameters have been restricted to ensure that they not only satisfy physical realism, but also maintain the look of plasticine materials. The user parameters refer to specific properties of plasticine, such as oiliness and surface roughness.

#### Parameterisation

It was important that the parameters for the model were intuitive for the user. Some time was spent assessing the key properties of plasticine and aligning them with the variables in the new model (equations 4.6 and 4.10).

In conversation with Aardman Animations it was indicated that a relatively restricted model with limited parameters would be desirable. One of the requirements was that the material could not be adjusted so much that it would no longer resemble plasticine.

The final parameters created for the plasticine model are as follows:

- base colour of plasticine
- transparency of the plasticine material
- oiliness: parameter to control the strength of the specular component of the material. From inspection it was noticed that plasticine produces more oil and becomes shinier when it is warm or if it is left for some time. This parameter allows artists to adjust the oiliness, so the plasticine can be designed to have a certain look, for example, to look like newly moulded plasticine.

- surface roughness: the amount of texture on the plasticine will affect the specular highlight. Some plasticine models are hand-smoothed more than others so this parameter allows that to be varied on the model.
- enforce energy conservation: enforcing energy conservation means that the additional changes to the model (section 4.4.2) are not used. It uses the combined model shown in equation 4.6 which will ensure that energy is conserved. This allows artists to use a physically-based model if they wish.
- grazing shine strength: this control is only available if 'Enforce Energy Conservation' is off. It allows control over the strength of the specular values created at grazing angles.
- grazing shine range: this control is only available if 'Enforce Energy Conservation' is off. It allows control over the range of values at which the grazing shine is added.
- sheen: this is an additional parameter visible on an 'Advanced' tab for the shader. It affects the behaviour of the specular highlight. It is the index of refraction parameter in the original model and changes the specular light reflection over the model.

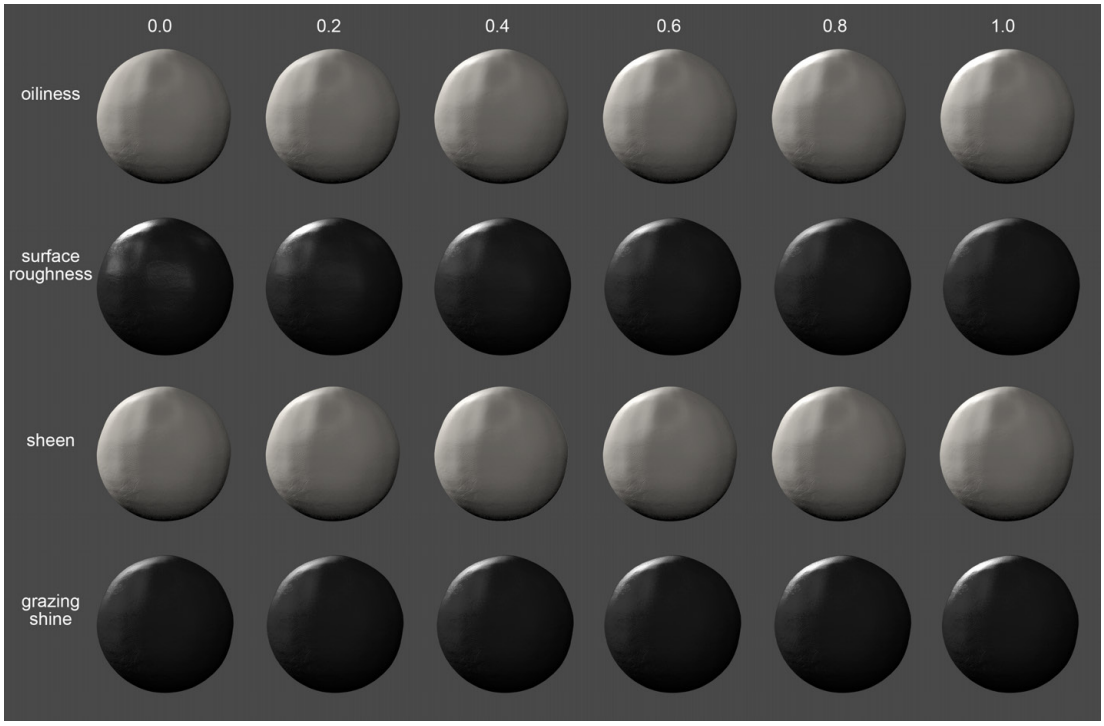



Figure 4-7: Examples of the parameter ranges for controls on the PlasticinePlus shader.

The range of the parameters was restricted so that they would only model realistic looking plasticine. The ranges of some of the controls are shown in figure 4-7. Acceptable ranges for each of the parameters were found with some parameters restricted based on the true physical properties of the material. For

| Model | Black Plasticine | | White Plasticine | |
|---|---|---|---|---|
| | Fitting error using least squares fit to the captured plasticine data | Comparison to the best fitting model prior to this research (Walter et al., 2007) | Fitting error using least squares fit to the captured plasticine data | Comparison to the best fitting model prior to this research (Walter et al., 2007) |
| Ashikhmin-Shirley model | 611020 | -72.91% | 452400 | -0.11% |
| Simple Blinn | 511000 | -44.6% | 420980 | 6.84% |
| Microfacet Blinn | 488300 | -38.18% | 615430 | -36.19% |
| Cook-Torrance model | 426370 | -20.5% | 543100 | -20.18% |
| GGX model | 353380 | 0% | 451900 | 0% |
| The PlasticinePlus model | **282550** | **20.04%** | 377190 | 16.53% |
| The Plasticine model | 283150 | 19.87% | **347230** | **23.18%** |

Table 4.1: The results of mathematical data fitting between analytical shading models and the recorded BRDF data. The error represents the minimised sum of the squares of the errors between the captured data and the optimised analytical model. The percentage improvement is calculated in comparison to the best fitting analytical model prior to this research, which is that of Walter et al. (2007).

example, the sheen parameter was restricted so that only realistic and physically likely values for the index of refraction for plasticine were allowed. Others were restricted based on what looked acceptable and retained the look of plasticine. All parameters were set within a range of zero to one for ease of use.

## 4.5 Experiments

A number of experiments were conducted to compare my plasticine shaders with state of the art shading models. I compare the results from mathematically fitting the models to the data captured in the BRDF table. Table 4.1 shows the resulting error from the data fitting. Figure 4-8 shows the visual results for the captured BRDF data, the two best fit existing shading models of Walter et al. (2007) and Cook and Torrance (1982) and the proposed models. Each model's parameters are the best fit parameters generated computationally. The discrepancies visible in the spheres modelled using the BRDF data are due to the texture information present on the captured plasticine.

The areas in which the new model improves the fit are subtle. To understand the areas of improvement in more detail I have analysed intensity profiles from spheres lit at varying angles. Figure 4-9 shows the intensity profiles with front-facing light and with light at grazing angles. The intensities in the images generated directly from BRDF data were smoothed using a moving average filter to reduce noise.

Tests were also completed in a more industrial context. Figures 4-10 and 4-11 show V-Ray renders comparing the benchmark model with the new PlasticinePlus model. Figure 4-10 shows the surface shaders when applied to a sphere. Sample displacement maps have been added to the spheres to demon-

strate the use of this shading model alongside texture information. Figure 4-11 shows the shaders when applied to a more complex character model.

### 4.5.1 Results

The results of the experiments show that the new plasticine shaders shows a significant improvement in simulating plasticine accurately when compared to existing shading models. The results of the mathematical data fitting (table 4.1) demonstrate that the models presented here show improvements of over 20% when compared to the next best fitting state of the art model.

The intensity profiles (figure 4-9) show that while retaining the accuracy of other methods in front-facing lighting, the PlasticinePlus model shows significant improvement in the modelling of specular peaks at grazing angles. The specular peaks generated using the PlasticinePlus model are stronger and wider which is more similar to the behaviour of the measured material (figure 4-9d).

Figures 4-10 and 4-11 show the improvements from using the new PlasticinePlus model in comparison to the V-Ray benchmark model currently used in production. Again, particular improvements are noticeable in the specular highlights when the light is at grazing angles. The new model mimics the oily sheen which appears on real plasticine more accurately, as visible when comparing figures 4-10c and 4-10d. The behaviour of the microfacet shadowing is also more accurate using the new model and represents the surface roughness of the plasticine surface. The results show a significant advancement and provide Aardman Animations with a bespoke plasticine shader for use in production, which has been demonstrated to be more similar to the physical properties of plasticine.

## 4.6 Discussion

The results show the improvements of these new shaders in accurately modelling the surface properties of plasticine when compared to state of the art and existing production solutions. Table 4.1 demonstrates that the Plasticine shader reduces the fitting error by around 20% or more for both black and white plasticine samples, when compared to the best fitting state of the art model. This is a significant improvement and helps to explain the improvement in realism seen in the image results (figures 4-10 and 4-11).

The new model particularly focusses on improving the simulation of specular highlights at grazing angles, as that is where other state of the art models fail to represent plasticine accurately. Figure 4-9 demonstrates the stronger and wider specular highlights exhibited in both real plasticine and by the PlasticinePlus model. The PlasticinePlus model retains the accuracy of other models at front-facing lighting (figure 4-9b) but significantly improves the fit at grazing angles (figure 4-9d).

It is a known problem that the recorded data becomes less reliable at grazing angles (Ngan et al., 2005). My captured data appears less reliable at these angles as the texture on the sample causes large fluctuations in the data (figures 4-9c and 4-9d). I used a combination of perceptual and mathematical fitting to develop a tool that is both physically accurate as well as artistically expressive. Attempting to mathematically fit to data with this level of noise would mean averaging out the specular peaks which was not desirable. Therefore, the additions made to the algorithm were based on visual inspection. The

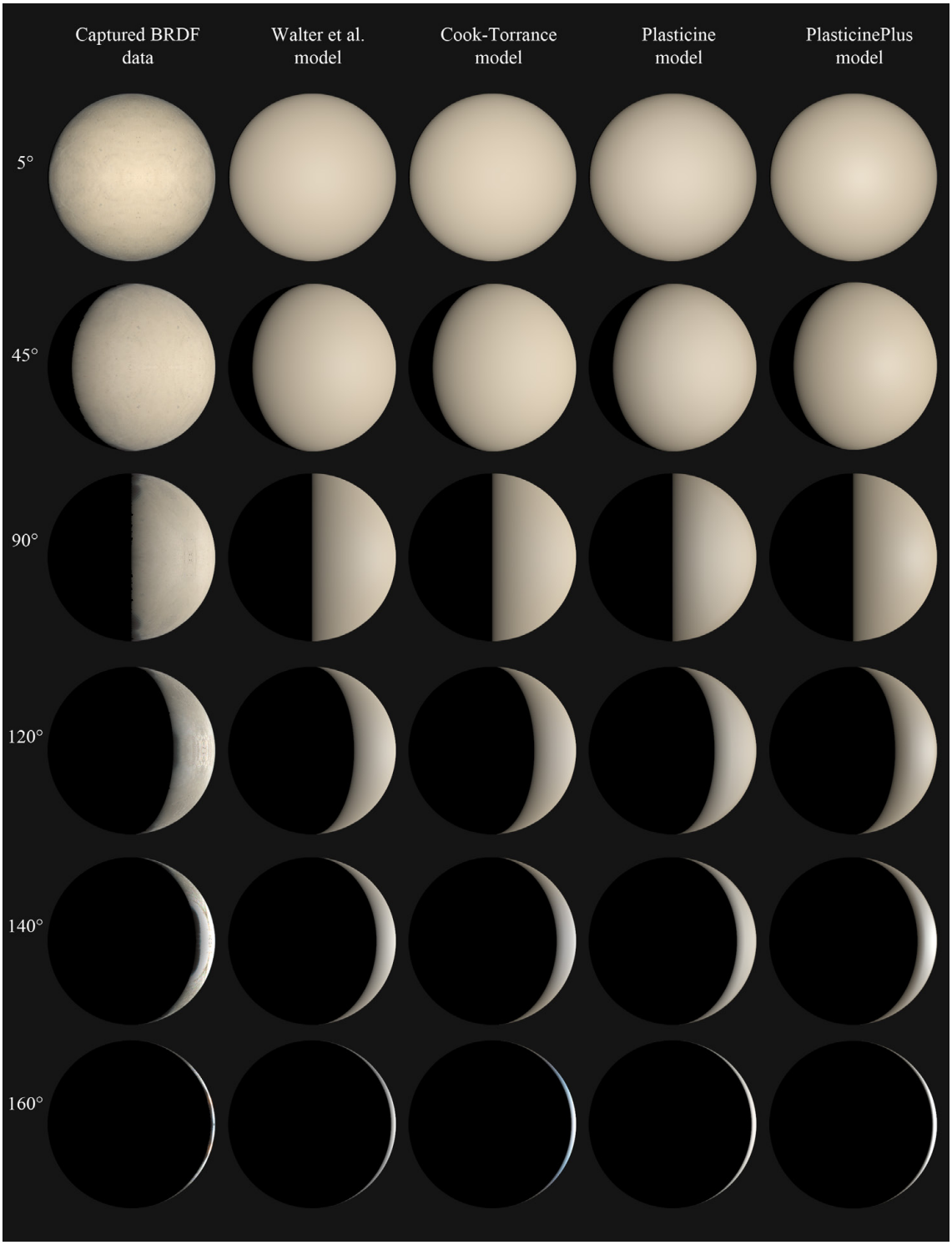



Figure 4-8: Spheres showing the two best fitting state of the art shading models and the proposed Plasticine and PlasticinePlus models, with incident light at $5°$, $45°$, $90°$, $120°$, $140°$ and $160°$. Each shading model is rendered using the best fit parameters generated by the mathematical data fitting. The differences between models are most visible at grazing angles ($140°$ and $160°$). (Figure continued on following page).

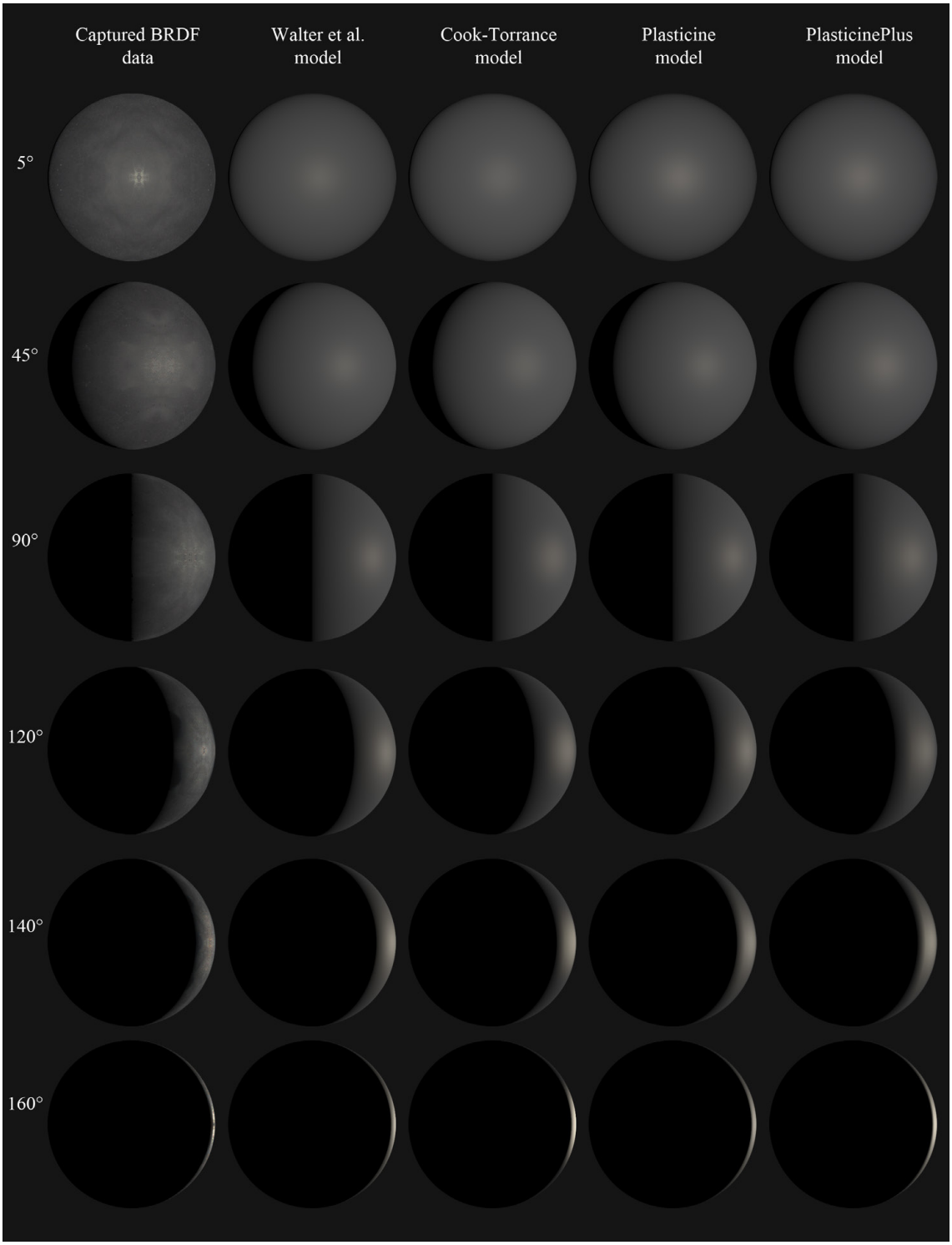
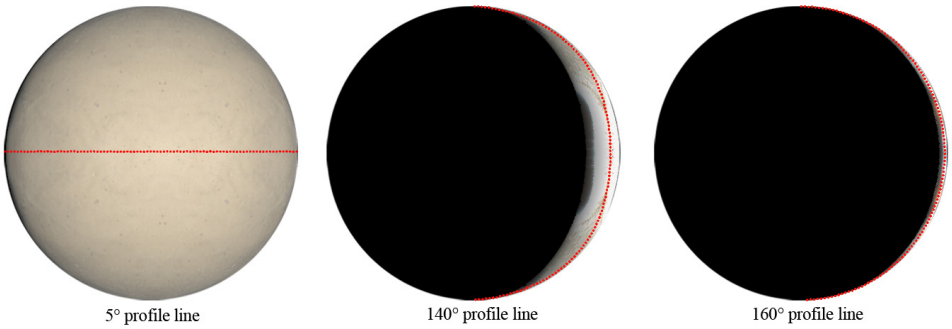
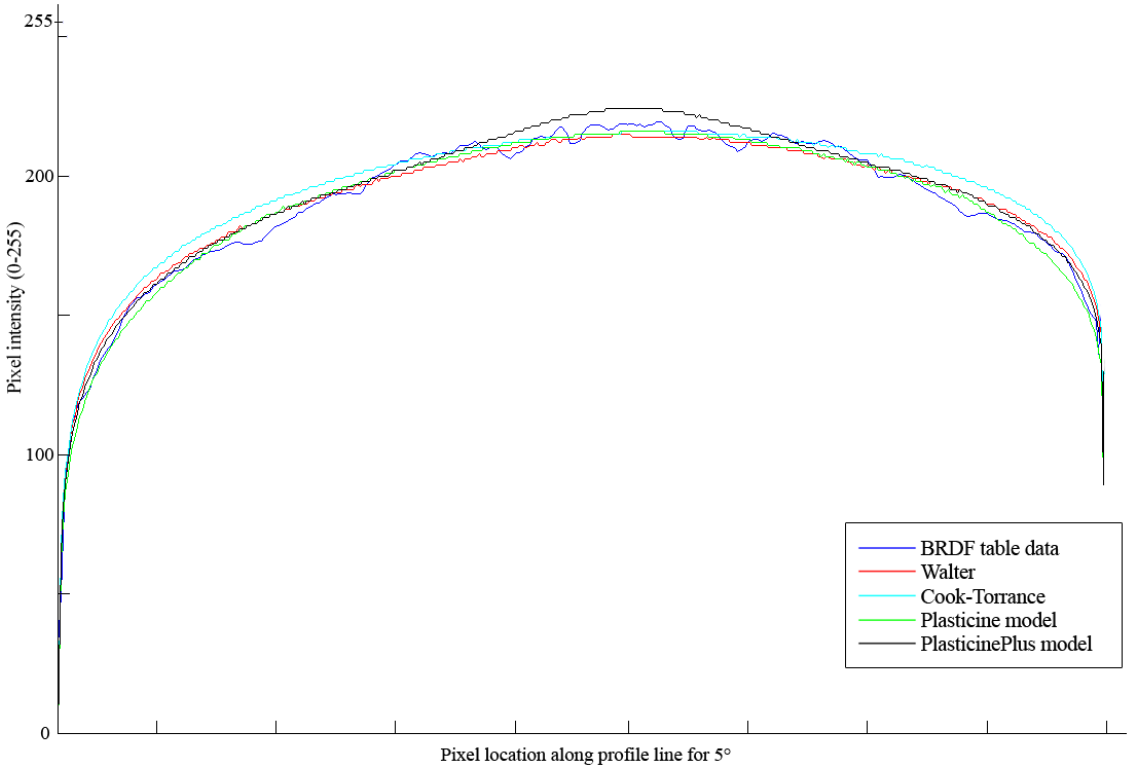



Figure 4-8: (continued) Spheres showing the two best fitting state of the art shading models and the proposed Plasticine and PlasticinePlus models, with incident light at $5°$, $45°$, $90°$, $120°$, $140°$ and $160°$. Each shading model is rendered using the best fit parameters generated by the mathematical data fitting. The differences between models are most visible at grazing angles ($140°$ and $160°$).

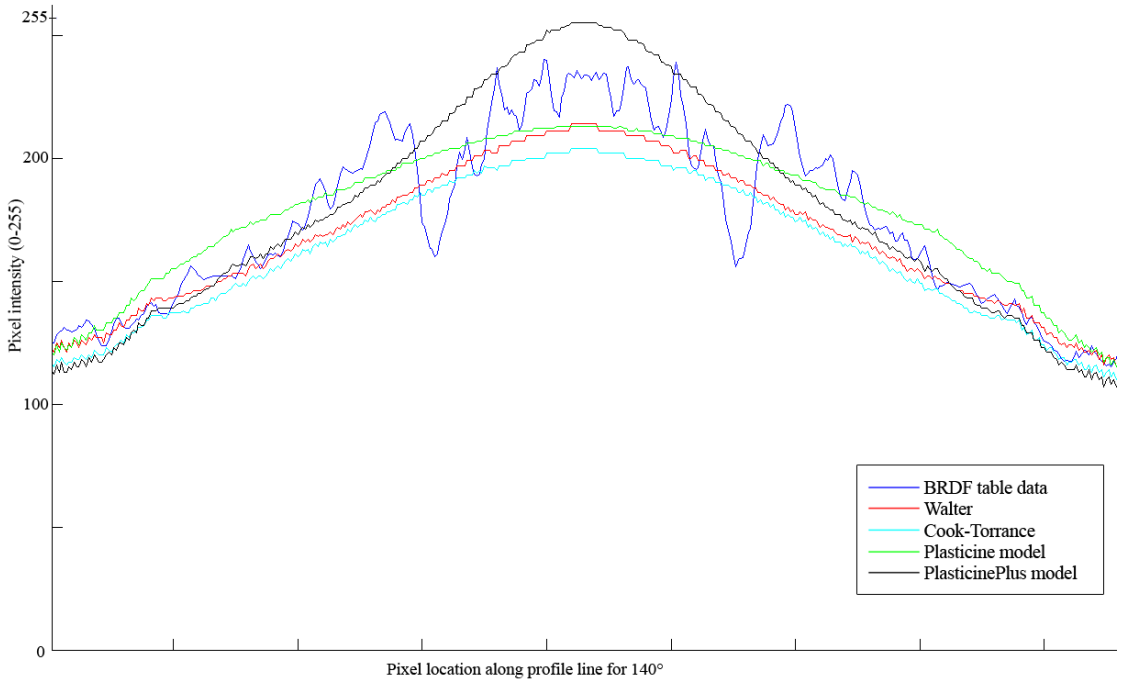



(a) Profile line positions used for the intensity comparison. The red line represents the pixel locations tested when generating the intensity profile graphs 4-9b, 4-9c and 4-9d.

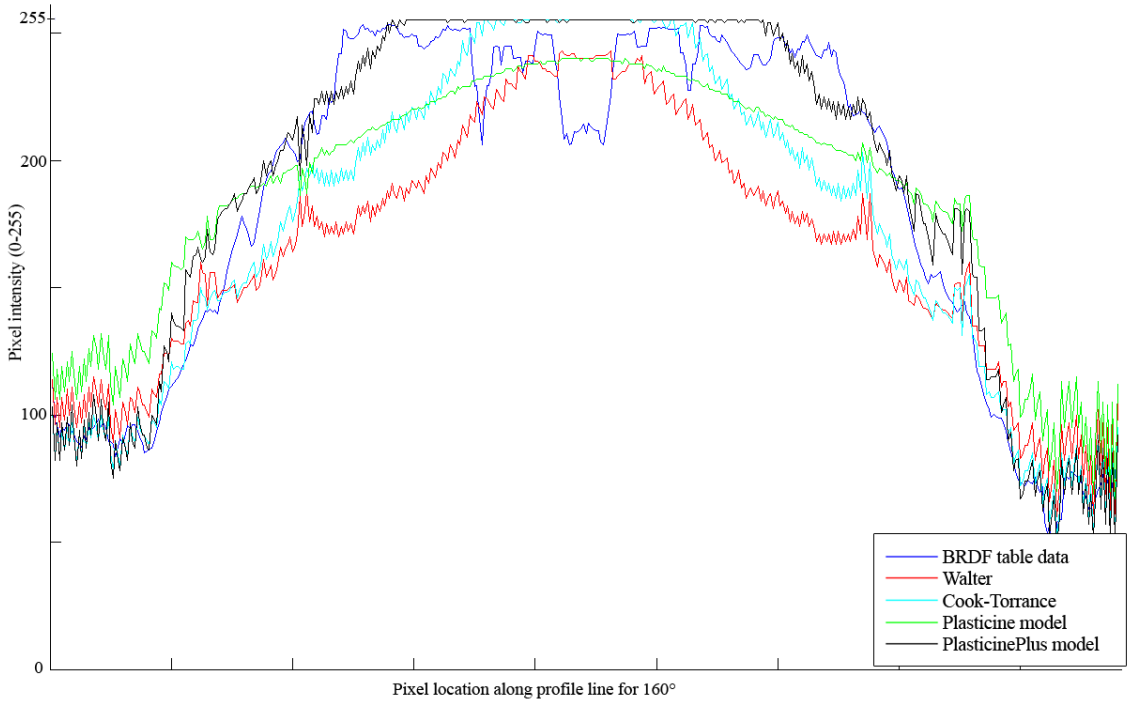



(b) Intensity profile at 5°

Figure 4-9: Intensity profiles at 5°, 140° and 160° comparing pixel values rendered using the captured BRDF data for white plasticine to those rendered using Walter et al. (2007), Cook and Torrance (1982), the Plasticine model and the PlasticinePlus model. The positions of the pixels sampled are shown in (a) (figure continued on following page).

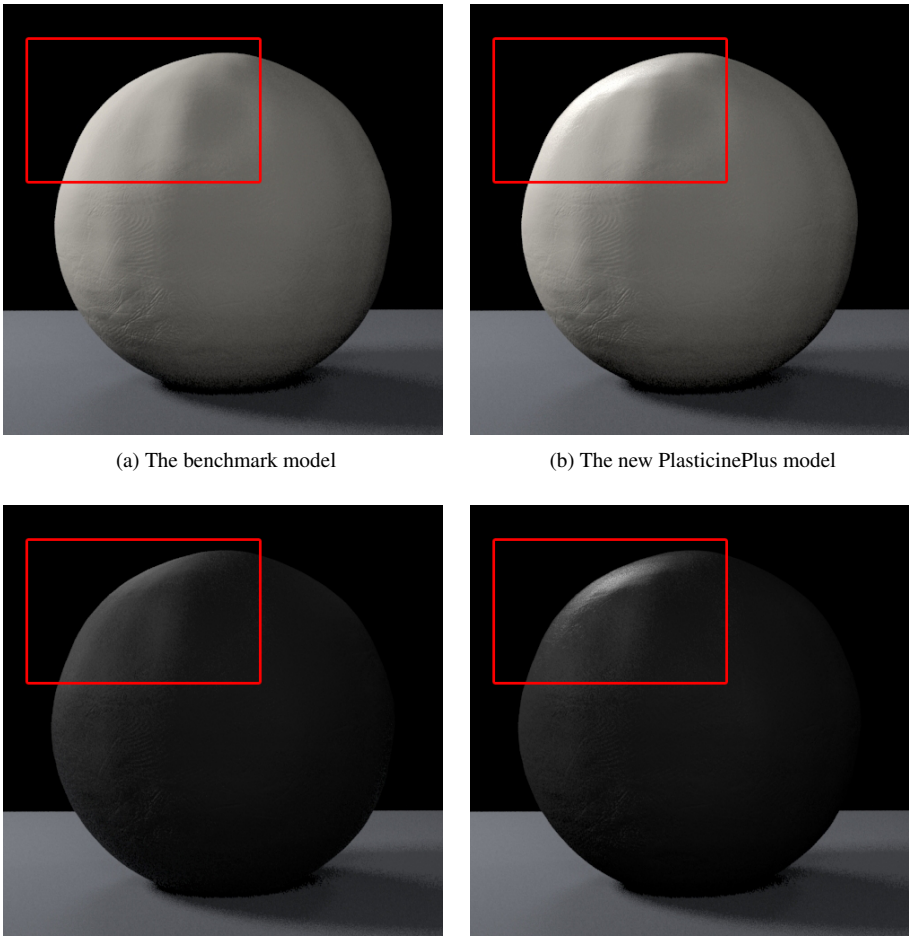



(c) Intensity profile at 140°




(d) Intensity profile at 160°

Figure 4-9: (continued) Intensity profiles at 5°, 140° and 160° comparing pixel values rendered using the captured BRDF data for white plasticine to those rendered using Walter et al. (2007), Cook and Torrance (1982), the Plasticine model and the PlasticinePlus model. The positions of the pixels sampled are shown in (a).



(a) The benchmark model

(b) The new PlasticinePlus model

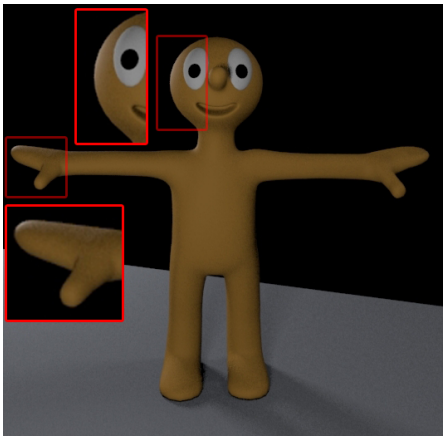
(c) The benchmark model

(d) The new PlasticinePlus model

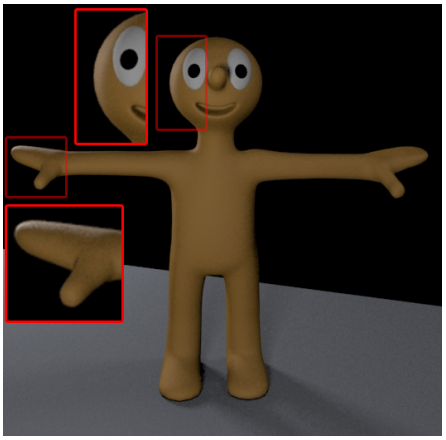
Figure 4-10: Comparison of the benchmark model ((a) and (c)) to the new PlasticinePlus shader ((b) and (d)) on textured spheres.

visual inspection involved comparing captured images to simulated results and comparing the graphical outputs of captured BRDF data with those from mathematical shading models. The results for the PlasticinePlus model in table 4.1 show that perceptual assessment does not always align with numeric quality.

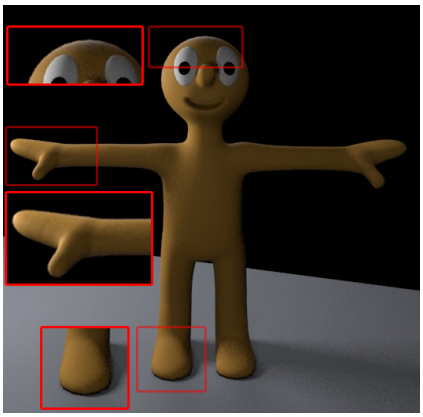
The results of the least squares data fitting, shown in table 4.1, show that black plasticine and white plasticine react differently to the two proposed plasticine models. For black plasticine the results for the Plasticine and PlasticinePlus models are very similar, with the PlasticinePlus model being marginally better at fitting to the data. For the white plasticine, the results show that the Plasticine model provides a significantly better fit. As discussed, the PlasticinePlus model was developed visually and for artistic control. The difficulties of recording data at grazing angles, the angles for which the enhancements of the PlasticinePlus model was designed, mean that an improvement in mathematical data fitting was not expected. Therefore it is not surprising that the PlasticinePlus model increases the fitting error in some cases. The reason for the white plasticine result showing a higher error than that of the black



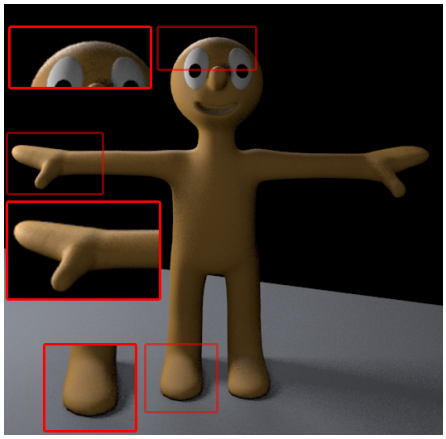
(a) CG Morph in lighting set-up 1: The benchmark model currently used in production



(b) CG Morph in lighting set-up 1: The new PlasticinePlus model



(c) CG Morph in lighting set-up 2: The benchmark model currently used in production



(d) CG Morph in lighting set-up 2: The new PlasticinePlus model

Figure 4-11: Comparison of the benchmark model ((a) and (c)) to the new PlasticinePlus shader ((b) and (d)) on a character model of Morph ©Aardman Animations Ltd 2014.

plasticine is thought to be due to the texture present in the recorded data. The shadows generated by this texture affect white plasticine more strongly as they create greater contrast in the surface colour of white plasticine than they do in that of black plasticine. These disparities in colour across the specular highlight mean that the fitting error is greater for a model that enhances specular highlights, such as the PlasticinePlus model.

Black and white plasticines were chosen for this initial investigation so that both light and dark varieties of the material could be tested and to ensure that any final developed models would not be colour specific. The difference in the results between the two varieties show that this was a valid decision. The fact that there are differences suggests that it would be worthwhile analysing a wider range of colours to test how the well the proposed models fit with a range of plasticine colours.

Texture information is very important in replicating the look of plasticine. As mentioned, some texture information was captured in the BRDF acquisition data. This does affect the data fitting and

the resulting error calculations. Table 4.1 shows that the error is large for all models when fitting to the recorded data and this is in part due to the noise added by the texture information present on the spheres. It would be interesting to test plasticine samples with minimal texture and compare results.

Figure 4-8 shows images created using each model with their calculated best fit parameters and it shows that some state of the art models produce a reasonable reproduction of a plasticine material in low-contrast lighting. However, it was found that without mathematical data fitting it can be difficult to get the same level of accuracy using the state of the art models. For example, when adjusting material parameters by eye, for these models it was difficult to find a configuration which represented plasticine well. My new shading model offers parameters and ranges set specifically for plasticine materials, as well as improved simulation of the material.

## 4.7 Creating a Texture Model

Using the surface shader generated in section 4.4.2, results were produced representing the reaction of plasticine materials to lighting. However, the material's behaviour is also affected by its texture. Due to the malleable nature of plasticine, there are many subtle indentations and textures in the surface which affect the way it reflects light. Therefore the next step was to look at simulating the textures of plasticine in order to generate a more realistic plasticine model to combine with the new lighting model.

### 4.7.1 Large Scale Displacement

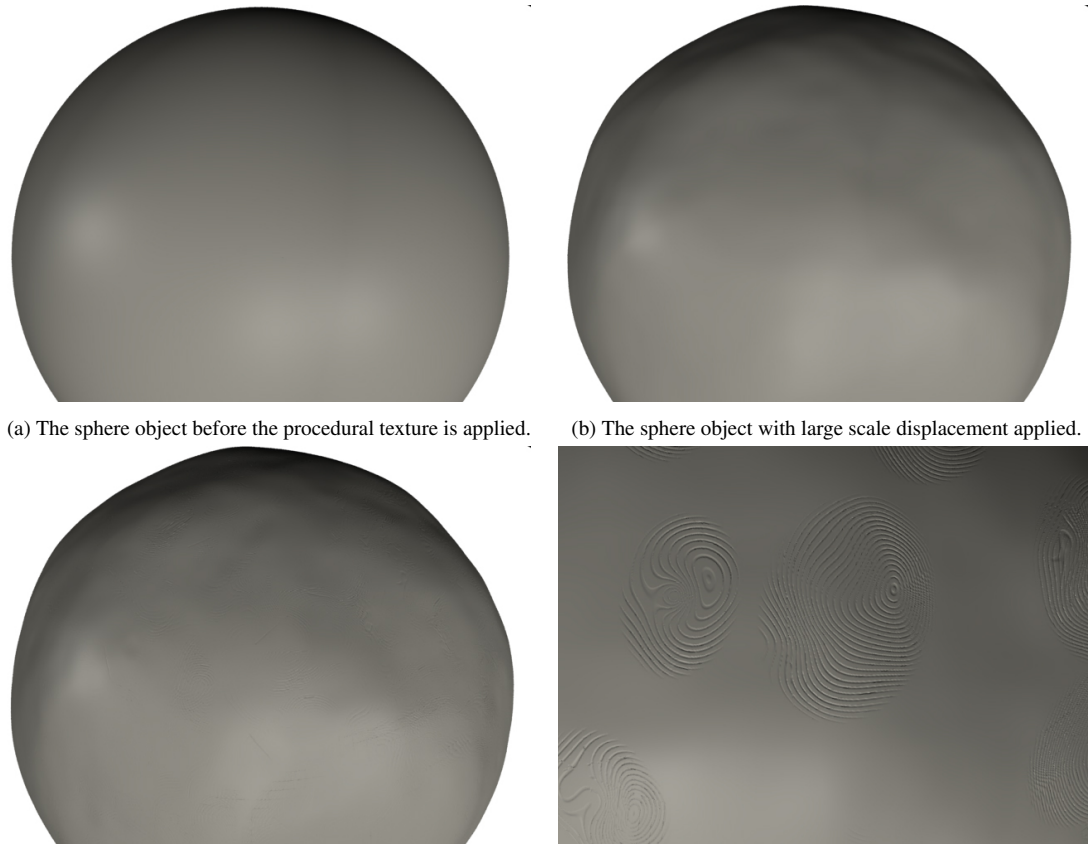
As noticed in the plasticine measurements shoot, it is impossible to get plasticine to form a 'perfect' shape of any type. Therefore, one of the first additions to the texture model was some large scale displacement to simulate the lumps and bumps which naturally occur when modelling in plasticine. To simulate this, Perlin noise (Perlin, 2002) was used across the surface at a number of different scales and frequencies. This meant that the surface looked uneven in a relatively random fashion (figure 4-12b).

### 4.7.2 Fingerprints

Having looked in detail at moulded plasticine, it became clear that it was the fingerprints and imprints of the modeller's skin texture that create the base texture on plasticine. There are other marks like small scratches and imprints from tools used, but the skin indentations are the primary surface pattern.

There were a number of ways that the fingerprint displacement could be modelling onto the plasticine model. The simplest way would be to use images of fingerprints as displacement maps and position them across the surface. However, this would require the management of a database of fingerprint images. Therefore it was decided that an alternative method would be pursued, in which the fingerprint textures could be generated procedurally.

In order to get the prints to appear at random positions on the surface of the plasticine a texture bombing technique (Trippe, 2009) was used. This 2-dimensional technique was extended to allow the modelling of 3-dimensional fingerprints. The reason for using 3-dimensional texture bombing is that the texture could be applied to the surface without needing to unwrap the model. Unwrapping is the



(a) The sphere object before the procedural texture is applied.

(b) The sphere object with large scale displacement applied.

(c) The sphere object with procedural fingerprints and scratches applied.

(d) Close-up of the procedural fingerprints before the directional fade is applied

Figure 4-12: Demonstration of the procedural plasticine texture when applied to a sphere.

process of mapping pixels in a 2-dimensional image to positions on the 3-dimensional object surface and can be a very time-consuming process. Using texture bombing means the texture can be applied without performing this task. Furthermore, texture bombing prevents problems with visible seams in the texture map and distortion in pinched or stretched areas of the mapping.

To create fingerprints, rings were created around the centre of the bomb. These rings were stretched so that they would appear as ellipses, with the direction of the stretch varied randomly. Perlin noise (Perlin, 2002) was added to rings so that they warped, with the noise scaled based on the distance from the bomb centre. This produced the random swirl pattern of a print but with consistent movement across the rings (figure 4-12d). The seeds for initiating the noise algorithm were generated from the position of the bomb centre so that the fingerprint patterns varied. The distances between rings was also varied by applying noise to the distance of the ring from the bomb centre.

The depth of each print was adjusted depending on the position of the ring on the finger tip, for example, central rings were imprinted more lightly. Additionally, a directional fade was applied to each print. This fade simulates the appearance of a print in plasticine which tends to be made from a particular direction. Figure 4-12c shows the procedural fingerprints when applied to a surface. The

addition of the fingerprint textures subtly breaks up the specular highlights, which makes the surface appear more matte. Adding this surface texture makes the simulated plasticine react to light in a way that is more like real textured plasticine.

### 4.7.3 Surface Scratches and Scuffs

In addition to fingerprints and surface displacement, research was completed into modelling other texture marks, such as scratches or scuffs. A similar approach was taken to that of fingerprints, by using 3-dimensional texture bombing to project the marks onto the surface of the object. Lines, or randomly spaced groups of lines, were created procedurally to simulate the scratches and scuffs on the surface.

Each of the textures detailed have a range of controls on the user interface. For example, the density, scale and strength of the procedurally generated fingerprints can be adjusted.

## 4.8 Conclusions

This research project has presented a bespoke shading model for plasticine materials. My novel algorithm shows a significantly improved approximation to measured plasticine data with a reduction in fitting error of over 20% when compared to state of the art shading models. Particular attention was paid to the accurate modelling of the specular highlights at grazing angles which contributes to the distinct sheen on plasticine materials. The aim of this research was to provide a more accurate model than the benchmark model which is currently used for computer-generated plasticine work. Figures 4-10 and 4-11 demonstrate the improved results.

The PlasticinePlus model has been implemented into production systems at Aardman Animations and offers a set of plasticine specific user controls to make the modelling of plasticine materials easier as well as more accurate. The user parameters allow for more versatility in the generation of plasticine materials. For example, artists have the ability to model different types of plasticine such as freshly moulded plasticine or older plasticine, which can develop an oily sheen.

It is likely that the improvements shown by the algorithms presented here are due to the scientific analysis of the physical properties of plasticine materials. For this reason it would be interesting to analyse the lighting model of plasticine in more detail and with more accurate acquisition equipment. The BRDF acquisition research conducted here was sufficient to give an approximation of the reaction of plasticine to light. However it was affected by some surface texture, as discussed. Use of a different acquisition set-up, such as one which uses a flat sample (Ward, 1992; Ghosh et al., 2010), would enable the generation of data for a smooth plasticine sample. This would provide a more precise measurement for the lighting model of plasticine without texture affecting the results. In this research it was assumed that plasticine is an isotropic material and it would be worthwhile testing this assumption. Using a perfectly smooth sample could establish whether plasticine is truly isotropic.

In addition to understanding the lighting model of plasticine, the texture of the material plays a very important role in its appearance. A perfectly smooth plasticine surface will not look like plasticine to a viewer, as most people are not familiar with seeing plasticine without marks and indentations. These textures affect the way that the material reacts to light along with the shading model. Investigation into

understanding and characterising the textural properties of plasticine has begun. Work has started into the development of a procedural texture model which can complement the shading model presented here. In future work it would be interesting to analyse textured plasticine with a scientific set-up (Glencross et al., 2008). The data gathered could be used to understand the textures typical to a plasticine sample and incorporate these findings into the procedural texture shader.

The basis of this research could be extended to other variants of plasticine. For example, plasticine which can not be created in the real world, such as patterned or translucent plasticine. There is additional data which could be collected using the BRDF acquisition system, such as different colours of plasticine. It would be interesting to see if these colours fit within the PlasticinePlus model or if different colours of plasticine exhibit different behaviours.

# Chapter 5

# Implementing in Industry

## 5.1 Introduction

This chapter details the additional work and requirements of producing research within an industrial environment. To complete academic projects in a way that is appropriate to industry, additional factors needed to be taken into consideration. While conducting industrially-based research it is important to be aware of the end goal of the company and the context in which a successful tool would finally be used. For example, integrating research results into the production pipeline is essential to transferring research for use in industry. The aim of each project was to produce tools which are accessible to and useable by production artists.

As part of producing research in an industry environment, the first task was to understand how research fitted into the structure of the host company (section 5.2). Additionally, it was important to gain a thorough understanding of the current production pipeline and processes. This was achieved by completing smaller development projects in the production pipeline, two of which are detailed in sections 5.3.1 and 5.3.2. Projects such as these helped to develop a knowledge of the software environments that would be needed in the larger research projects.

The requirements and considerations of working on the larger research projects of rig removal (section 5.4), set shift (section 5.5) and plasticine shading (section 5.6) are detailed. This is followed by discussion about the considerations and challenges of conducting research in an industry environment (section 5.7).

## 5.2 Understanding the Industrial Landscape

The structure and interests of a company affect how academic research can be integrated into production. In the early stages of research, it was important to understand that structure at Aardman Animations and how they intended academic research to fit into their business. Pertuzé et al. (2010) conducted research on industry-university collaborations and found that "research is more likely to have positive impact on a company if the university researchers have strong knowledge of the business setting, company

practices and how the research fits company strategy".

Aardman Animations do not have a dedicated research department so they wanted research to be conducted in one of their existing departments. The department interested in being involved in additional research projects was the CGI (Computer Generated Imagery) department. The CGI department predominantly work on commercials, but are also involved in short form animation and some television and film work. They work with both stop-motion and computer generated content. Footage for stop-motion commercials is processed by the CGI department for post-production work. Aardman Animations is a creatively-driven company and it was clear that technical research must not conflict with the traditional creative ethos of the company. The key goal is to aid the creative process, using computational tools to enhance the existing methods.

One challenge of not being part of an existing research department is that there is little precedent for conducting research. As the company is creatively driven, there is no established research plan in which to integrate the research projects. Typically research work within this department has been production-led and relatively short-term when compared to the timescales of academic research. For example, in production a few weeks might be spent researching and developing a new plug-in which is required for a particular effect on an up-coming commercial project. This meant that serious consideration had to be given to finding relevant research projects that would satisfy both an industrial need and the novel contribution required by academia.

## 5.3 Understanding the Production Pipeline

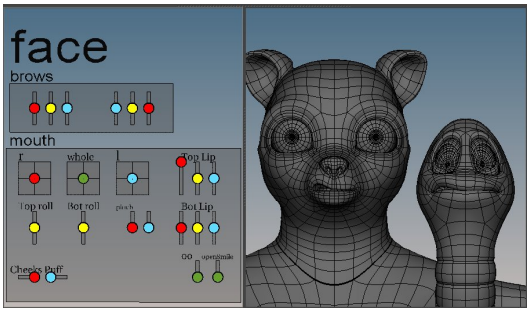
It is important to understand the structure with which final developed tools must integrate. The production pipeline outlines the stages of a production project and the processes which must be completed to produce the final deliverables. Therefore, a thorough understanding of the production pipeline is vital. It is also important to understand how users currently approach and solve each of the research problems to be investigated, to gain an understanding of what a new tool would be compared against.

The aim of each research project was to create tools for use in production. With this comes the issue of rolling out developed tools into the production environment for testing and use on production. Aardman Animations has two environments that can be used for running production processes, the live environment and the development environment. All tools were written and tested in the development environment, to ensure that they would not adversely affect the live production environment when transferred. Once testing in development was completed, tools could be rolled out to production for users to access and test. Each iteration of fixes or amendments used this roll-out process until the tool was finalised.

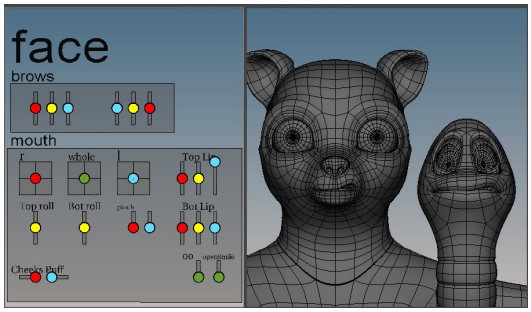
In order to understand the production pipeline at the company, I was introduced to some smaller development problems. Completing smaller research projects within the production pipeline provided me with knowledge of writing plug-ins for production specific software and the related application programming interfaces (APIs). Additionally, it provided an insight into the structure of the production process and the pipeline through which animated content is created. Two of the smaller development projects completed were the BlendshapeSplitter plug-in for Maya and the TimeSwitch plug-in for Nuke.

### 5.3.1 Maya: Blendshape Splitter Plug-in

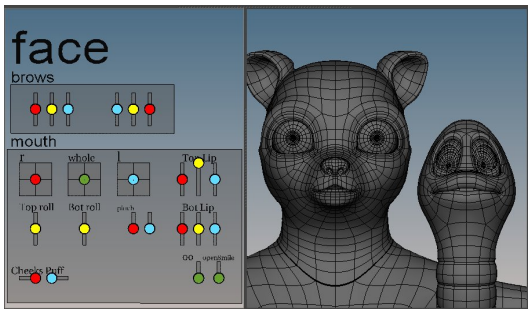
Autodesk Maya is used for modelling, rigging and animating computer generated content. Maya is widely used in the animation industry and is an integral part of production at Aardman Animations. Maya has an extensive API which can be used to write scripts and plug-ins to customize the application. In order to understand the inner workings of the software, I undertook some smaller research problems that would require implementation using the Maya API. One of these such projects was the BlendShapeSplitter.




(a) Left blendshape for the raised upper lip.




(b) Right blendshape for the raised upper lip.




(c) Central blendshape for the raised upper lip.

Figure 5-1: Resulting blendshapes after using the BlendshapeSplitter plug-in ©Aardman Animations Ltd 2014.

In animation, blendshapes are used to transform the points in a mesh into a range of positions. A blendshape is a weighted linear interpolation between two mesh shapes, with the points in the mesh moving into the positions dictated by the weighting. For example, for facial animation, a character's mouth might be modelled in a neutral position and in a full smile. The blendshape can use these two positions as inputs and output combinations of the positions from both expressions. The controls on the blendshape dictate exactly what combination is output. Creating the blendshapes for a character is a sizeable job as a character will often need many different facial expressions. Shapes need to be created so that every expression required in the shot can be generated.

Often it is required to have separate controls for the left and right hand side of the character's face and there are existing tools that can be used to split blendshapes into the two halves automatically (Supercrumbly, 2004). However, Aardman Animations required a tool that would not only split blendshapes into the left and right side but also have a controllable central blendshape generated automatically.

The new BlendShapeSplitter plug-in uses two shapes as input to generate three blendshapes for the left, right and central sections. Figure 5-1 shows example output from the Blendshape Splitter, showing the results after splitting the top lip into three separate blendshapes. For each character shown in the figure, two different face meshes were taken as input: the neutral face of the character with its lips together and the face of the character with its whole top lip raised. The blendshape splitter was run on the two meshes and it outputs three blendshapes for each character, allowing the left, right and central sections of the top lip to be controlled individually. The plug-in reads in the geometry of each mesh and separates them into the three sections. User controlled parameters determine the size of the central blendshape and its fall-off to the left and right blendshapes (figure 5-2).

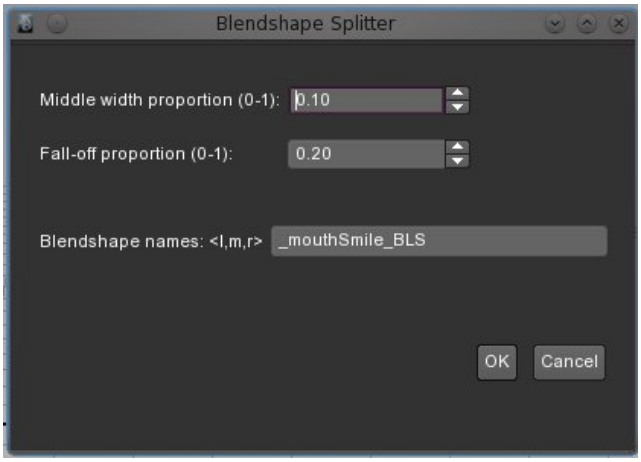



Figure 5-2: The BlendshapeSplitter user interface.

This plug-in is a useful tool which saves the modeller time. It is currently used in production and has reduced the time spent creating blendshapes for facial animation. “We had tools to split 2 ways but our facial animation system uses 3 controllers (left, middle, right) for each lip. It has proved to be a very useful tool as it quickly splits the expression, naming the new models in accordance with our naming conventions which can then be tested on the rig.” (Spence, 2014).

### 5.3.2 Nuke: TimeSwitch Plugin

Nuke is a popular post-production tool and commonly used in the animation and production industries. Aardman Animations use it extensively for post-production work and for this reason it was important to develop an understanding of the software. Nuke provides a developer kit, the NDK (The Foundry, 2014b). It enables the development of custom plug-ins to generate bespoke tools. Nuke plug-ins must be developed using a particular structure that is specified in the API. There are a number of different functions which must be used and which are run at different stages of processing. For example, the `engine()` function is called for each pixel in the input image as it is processed in a line-scan method.

In order to gain an understanding of the Nuke NDK I undertook a number of smaller Nuke plug-ins that required researching. One of these plug-ins was the TimeSwitch plug-in. The TimeSwitch plug-in was created for use with stop-motion footage. Like a lot of animation, stop-motion footage is often

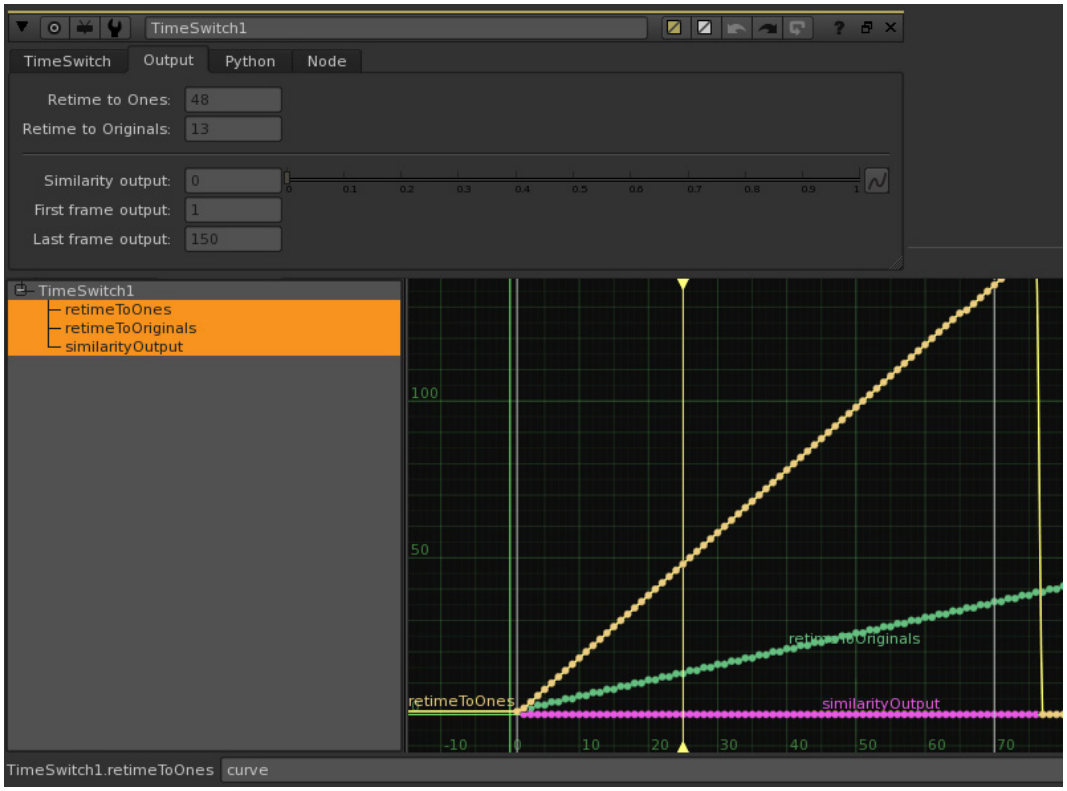



Figure 5-3: The TimeSwitch plug-in showing the 'Retime to Ones' and 'Retime to Originals' controls that are populated when the executable is run and their values in the graphical display. ©Aardman Animations Ltd 2014.

captured by replicating frames multiple times. Common forms are animating 'on ones' or 'on twos', where 'on ones' a shot is taken for every frame and 'on twos' a shot is taken every two frames with each frame duplicated. However, the timing may be varied throughout the shot without a pattern, with the number of repeated shots varying throughout the sequence. When there are duplicated frames, post-production work only needs to be completed on the initial image and the work can be duplicated for the repeat frames. For this reason post-production requested a plug-in that would temporarily remove duplicate frames, along with the functionality for the duplicate frames to be replaced after processing, to revert to the original timing.

The TimeSwitch plug-in was designed to check for duplicate frames by analysing pixels for similarity. The plug-in is a Nuke executable which means it runs as a single process, taking a sequence of frames as an input. The input frames contain duplicates and the desired output is a sequence with no repeat frames. The user interface contains a threshold control which dictates the allowable difference between frames, so that the user can account for camera noise in the images. When the program locates duplicate frames it records a frame's original position in an uneditable user control, 'Retime to Originals' and another uneditable user controls records its 'Retime to Ones' frame position (figure 5-3). The compositor can use the values in the 'Retime to Ones' control to connect to a Nuke TimeWarp node. The TimeWarp node takes these frame numbers as input and adjusts the timing of frames using the

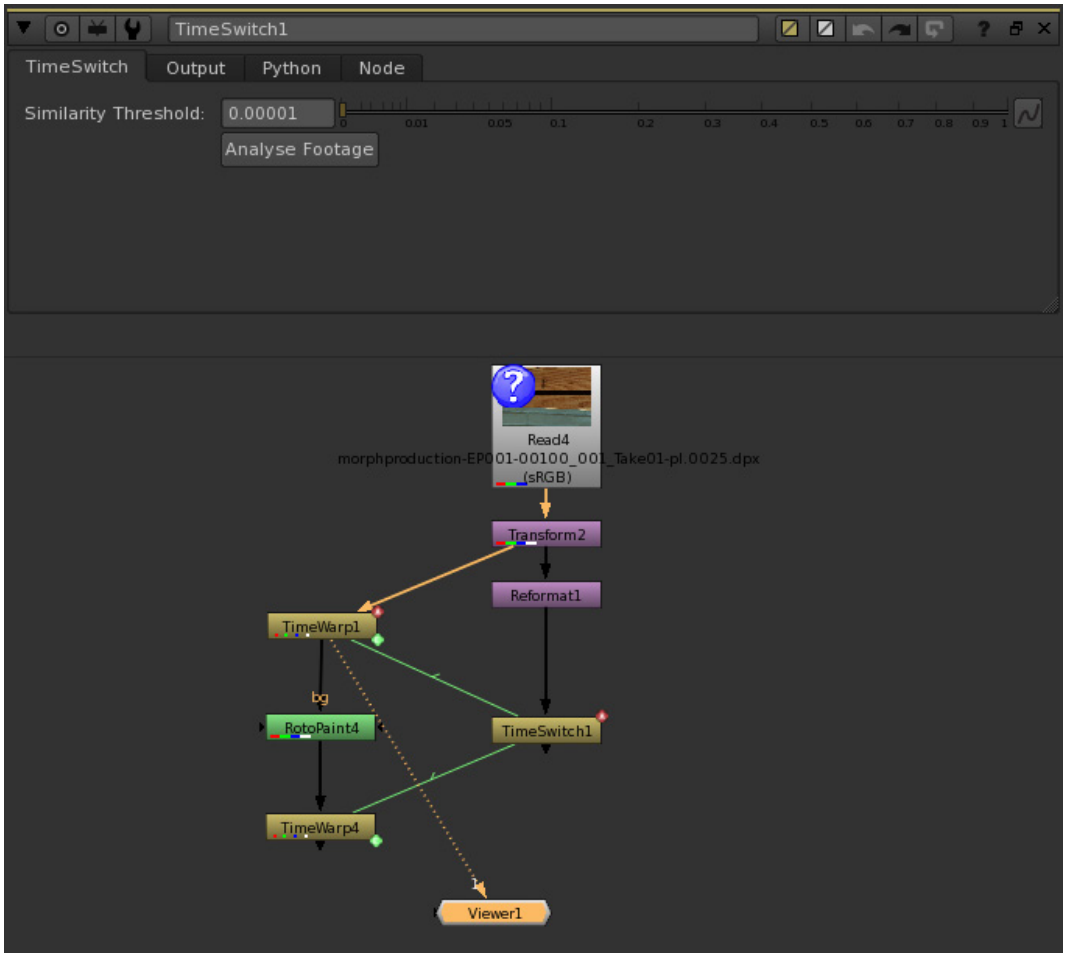



Figure 5-4: The TimeSwitch plug-in showing the user interface and the TimeSwitch node connected to TimeWarp nodes ©Aardman Animations Ltd 2014.

value in the control. The frames will then be displayed without duplicates and the user can then make the required changes to each individual frame.

Once the work on frames has been completed, the original timing of the frames needs to be reproduced. This is done by linking the values in the 'Retime to Originals' control with another of Nuke's TimeWarp nodes (figure 5-4)

This tool saves time in post-production as users can work directly on the unique frames and then their changes are duplicated, rather than having to copy changes across repeated frames. This tool is currently used in production and reduces the time spent when compositing shots with duplicate frames. "TimeSwitch is an incredibly useful tool for dealing with stop motion animation jobs. It makes the process of de-rigging and cleaning up the animation much quicker (up to twice as fast on some shots) and much more enjoyable for the operator. It's also incredibly quick at both performing the analysis, and in utilising Nuke's built in TimeWarp nodes, has practically no effect on processing speeds once it's all set-up." (Biggins, 2014).

## 5.4 Rig Removal

An initial requirement of the segmentation tool was that it would need to work in Nuke so that it would fit within the existing production process. For this reason, development work was completed in C++ and within the Nuke plug-in framework.

### 5.4.1 Implementing a Benchmark Tool

Initially in the rig removal project, an implementation of the GrabCut algorithm (Rother et al., 2004) was developed as a Nuke plug-in. This was done to test the level of results achievable using a state of the art segmentation algorithm. The GrabCut algorithm was integrated into Nuke using an Image class to hold the pixel information available from the Nuke interface. The implementation was adapted from an available C++ implementation (Talbot, 2010). Integrating this algorithm into a Nuke plug-in meant that user input could be added via the Nuke interface and the algorithm could be easily tested on production images.

The results from the GrabCut implementation showed some problem areas when tested on production images. Firstly the production images were large which made the algorithm very slow. Secondly, the accuracy was not to the required standard and thirdly, the algorithm was designed for single images so there was no temporal consistency across the video footage. Therefore, an alternative method of segmentation was researched.

The geodesic segmentation method detailed by Criminisi et al. (2010) looked to improve upon some of the problem areas demonstrated in the GrabCut algorithm. It claimed to be much faster while being at least as accurate. It was designed to work with image and video content which would help with the problems of temporal consistency. For this reason the geodesic segmentation algorithm was implemented as a Nuke plug-in. The implementation showed improvements on the results obtained using the GrabCut algorithm and for this reason it became the benchmark result. The geodesic segmentation tool also exhibited some inaccuracies when segmenting stop-motion footage. It was for this reason that the new algorithm was developed (chapter 2). Along with improving the algorithm, consideration had to be given to certain industrial requirements such as integration. The improved algorithm was implemented as a Nuke plug-in, which involved additional complexity in terms of processing the algorithm from within Nuke. The source code for the main Nuke plug-in class is shown in Appendix B and demonstrates how the algorithm runs from within the Nuke API.

### 5.4.2 User Interface

While working on the rig removal project, a lot of thought was given to the potential users of the tool. Having researched the state of the art in segmentation, it was known that the tool would need to be interactive and that users would have to indicate the area for segmentation. The GrabCut algorithm used a bounding box to highlight the object to be segmented. However, this was not useful in the context of rig removal due to the complicated shapes of rigging and characters.

It was suggested by post-production users that it would be more useful if the interface could take any rotoscoped shape as input. Therefore the bounding box was adjusted to take any shape as input to

allow flexibility when outlining the object area.

Later in the research, a different image segmentation tool, After Effects' Rotobrush (Adobe, 2012) was highlighted as one that had a user-friendly, stroke-based interaction style. It was suggested by post production users that this style of interaction was favourable when working on segmentation tasks and would be preferable to the bounding shape approach. The stroke interface was one of the positive attributes of the geodesic segmentation research (Criminisi et al., 2010) and contributed to the choice to implement the algorithm.

### 5.4.3 Speed and Optimisation

As the brief dictated that an interactive tool was required, it was important that the developed tool would react in a timely fashion to user input. This added a challenging additional requirement to the development of new tools. In analysing existing research I found that much research into video segmentation is focussed on the quality of the final results. Although time might be analysed, many segmentation algorithms are not fast enough to be used interactively in a production environment.

One of the drawbacks of the GrabCut implementation (Rother et al., 2004) was its speed to process production frames. When the algorithm was tested it was found to be much too slow to be used interactively on production size images, with speeds of about 20 to 30 minutes to process one frame in some cases. Additionally GrabCut was not designed for video footage so there was no temporal consistency. Criminisi et al. (2010) claimed that their geodesic algorithm was significantly faster than graph-cut based algorithms and that it worked with both image and video footage.

The new segmentation tool developed in this portfolio shows improved segmentation results. However, the tool was developed as a proof of concept so it requires further work for it to be useable as a production tool. In order for it to be an interactive, responsive tool it needs further development work to optimise the code. The industry aim for a useable tool was that it would run faster than the methods artists used to complete segmentation tasks.

An additional restriction on the speed of the tool is the requirement that it must be implemented into the production pipeline. This affects the speed as a developer has less control when optimising the code. As the code has to be integrated into the Nuke NDK, there is a large amount of code which is not under the control of the author. Therefore, performing certain types of operations within a Nuke plug-in might be slower than if you were to perform them in a standalone application.

### 5.4.4 Prioritising Research Projects

The aim of this research portfolio was to investigate problems in the production of stop-motion content. The company decided that research into rig removal was no longer the primary research aim of the department. Having completed initial research into this area and developed an improved algorithm, it was decided that this was as far as the rig removal project could proceed. The results indicated that the level of improvement needed in this area was extensive, in order for a production standard tool to be developed to the required accuracy.

Additionally production software had improved during the course of conducting this research. So the importance of researching in this area had diminished as other tools became available (The Foundry,

2014a). For these reasons it was decided that subsequent research work should be directed at different areas of the stop-motion production process.

This change in projects and priorities is insightful in understanding how research is conducted in industry. Aardman Animations tend to work in a flexible way with their research projects and prioritise their most pressing research issue. If it is discovered that a research project can not produce the required result in the time allocated, it is recommended that research resources are redirected to other areas where there is more potential for progress.

## 5.5 Set Shift

This research project needed to produce a tool to fix the problem of set shift, that would not require further tweaking from the user. One of the restrictions imposed on the set shift project was to avoid segmentation research. Segmentation had already been studied in detail during the rig removal project. It was known that the accuracy achievable was not to the subpixel level which is required for the set shift problem. For that reason the set shift research was restricted to start with a rotoscoped area and concentrate on producing a successful warp tool to correct the shift. The experiences with the rig removal project meant that I already had an understanding of the quality of results expected from an industrial post-production tool.

### 5.5.1 Prototyping an Application

As the feasibility of this project was unknown, a prototype application was built to assess the type of results achievable using state of the art research. Although a final tool would need to work in Nuke, building a prototype application was an easier approach than working directly in the production software in the design stage. Working on a standalone application allowed for rapid prototyping of ideas and algorithms. This saved development time and allowed for results to be generated quickly. Had this project proceeded to the development and implementation stages it would have been important to integrate any tools into the production pipeline in the same way that the segmentation tool was implemented.

### 5.5.2 Industrial Research Strategy

The results of the investigation showed that state of the art research in computer vision is not sophisticated enough to detect the very subtle shifts and warps that develop in stop-motion footage. Although these findings are not an academic advancement, from an industry point of view this is an important result. Aardman Animations are now aware of the intricacies of the set shift problem and understand it more fully. It was important to test state of the art implementations and to see the disparity between those results and the results required by industry. It was clear that the gap between current research and what was required by the company was too great. It would not have been possible to develop the required tool to a level that would satisfy the requirements of the company within the time that was available.

### 5.5.3 Prioritising Research Projects

The company felt that because the results from the prototype application were far from their required output, that reporting on the complexity of the task was a sufficient conclusion. It would not have been possible to develop a production tool given the time and resources available. This provided an interesting insight into how research that does not provide an industrial deliverable is considered valuable work in industry. Aardman Animations are satisfied with understanding the complexities and challenges of the set shift problem. This knowledge means that they will not spend time investigating possible solutions until significant advancements have been made in the associated computer vision research areas, such as subpixel edge detection or segmentation.

## 5.6 Plasticine Shading

The aim of the plasticine shading project was to produce a bespoke surface shader which represented plasticine materials accurately and was intuitive for artists to use. This meant that there were several restrictions on the research, such as that the shader would need to be developed to work with the company's current rendering software. For Aardman Animations this meant that the final surface shader had to be a V-Ray shader for Maya (Chaos Group, 2014). V-Ray is a rendering engine which outputs 3D modelled and animated content as image files.

### 5.6.1 BRDF Acquisition

Acquiring the BRDF data in an industry environment meant that the experimental set up needed to be possible using equipment available in the company. Fortunately, many of the required components were the same as that used for production tasks. A relatively simple BRDF acquisition experiment was chosen (Günther et al., 2005), so that components could be sourced within the company. Adjustments were made to the experiment based on the equipment available, such as the use of a CMOS sensor camera in the place of a CCD sensor camera.

### 5.6.2 Software Choices

Although the final shader needed to be developed for V-Ray, it was known that this was not an easy renderer in which to develop shaders, due to the lack of documentation and development resources. For this reason initial shading work was completed in Renderman Shading Language (RSL). Using this software provided a much more accessible and well-documented language that could be used to understand the theory of writing shaders. It was also a much quicker environment in which to develop shaders.

Writing the shader for V-Ray meant that a thorough understanding of the V-Ray software developer kit (SDK) was required. As there is limited documentation on writing V-Ray shaders, research had to be completed in order to create my own material and texture shaders. Once the structure of the V-Ray API was understood, ideas generated in Renderman shading language could be translated to V-Ray. As the structure of an RSL shader is different to that of a V-Ray shader it was necessary to adjust the

programming of the shader to accommodate for the differences in renderers. For example, when writing a displacement shader in Renderman, it is possible to move the point which is being shaded to create the displacement, whereas in V-Ray, the displacement must be created by outputting a greyscale value which becomes the height map for the displacement.

### 5.6.3 V-Ray Shaders

When writing the Plasticine surface shader for V-Ray, it was necessary to create a V-Ray material class and a V-Ray BRDF class. These two classes are used together to output a surface shader. The material calls the BRDF class which contains the mathematics to calculate the ratio of light returned at any combination of light direction, view direction and surface direction.

It was important that the shader was easy to access and looked familiar to users. A Maya plug-in was written so that the material would be visible when using the 'Hypershade' window in Maya, which is the window used when selecting materials to apply to models. The Maya wrapper plug-in is linked to the V-Ray surface shader plug-in. Adding a Maya wrapper to the V-Ray material also allows for customisation of the shader's user interface. Each parameter's range can be specified as well as the layout of the user controls and tabs.

### 5.6.4 User Interface

As this tool was to be used by artists and fit within the production pipeline it was important to create a tool that would be intuitive to use. As this shader is based on a physical model, the initial parameters were not easy for an artist to use. Some time was spent understanding the physically-based parameters so that they could be described in a user-friendly way. The parameter names shown in the plug-in are designed to be intuitive to artists and allow easy manipulation of the material. The names were adjusted to reflect key properties of plasticine such as oiliness and surface roughness. Additionally some of the less obvious parameters were added to an 'Advanced' menu so that they did not clutter the initial interface (figure 5-5).

The parameters were clamped so that the controls could not be used to create an unrealistic looking plasticine. Using trial and error experiments, acceptable parameter ranges were found. It was important to ensure that artists had the flexibility to create a range of plasticine looks, without being able to make it look unlike plasticine altogether. Once the parameter ranges were calculated, all were set to span from zero to one for ease of use.

### 5.6.5 Texture Generation

The work on the texture generation for plasticine materials also needed to be integrated into a V-Ray shader, by using the V-Ray Texture class. The procedural texture generation code was initially written in Renderman shading language due to the more simple API. Once ideas had been tested in Renderman, the code was translated to a V-Ray texture shader. In moving the code to V-Ray I also had to translate methods used in Renderman. For example, Renderman shading language provides in built functions

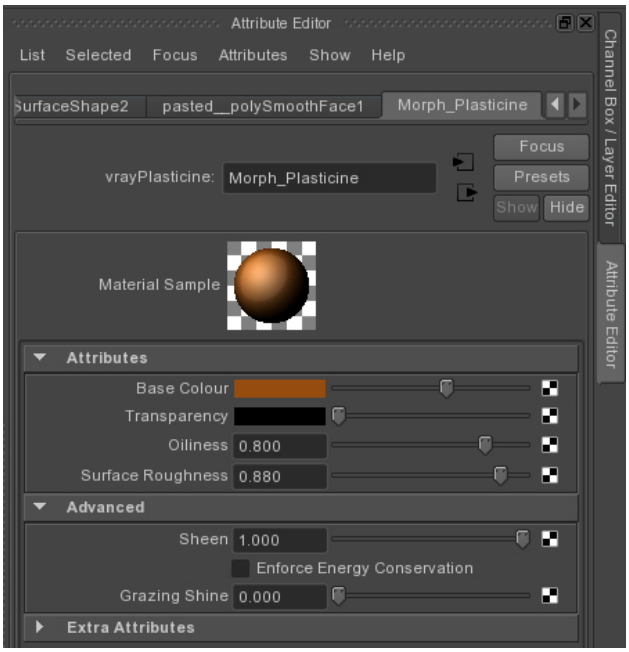



Figure 5-5: The Plasticine shader user interface.

such as `noise()` and `random()`. Whereas for the V-Ray texture plug-in, functions such as Perlin noise (Perlin, 2002) had to be implemented.

### 5.6.6 Integrating with V-Ray Version Updates

As tools such as V-Ray are often updated with new versions in an industry environment, it was important to ensure that the new shader would continue to work when new versions were installed. There was no existing infrastructure for the integration of custom V-Ray shaders so a system had to be developed for their roll-out. Links were set up with environment variables to ensure that each new version of V-Ray would continue to find the new plasticine shaders and that they would not be overwritten by new installations of V-Ray.

## 5.7 Discussion

Implementing research in industry meant that additional considerations need to be made throughout the research. Some of the additional requirements of developing this academic research in an industrial environment have been:

- Additional level of detail in the problem specification and user requirements.
- Integration into production software.
- Intuitive user interface.

- Additional testing time.
- Additional software support.

I found that working in industry, each project had a very specific end goal. This was positive in that it ensured that I kept focussed on the end requirements whilst conducting research. However, it also meant that research was restricted by the limitations imposed by industry. To understand these limitations at the beginning of any of the projects researched, time had to be spent researching the problem and the way in which the artists currently solved it. It was important to talk to the artists to understand exactly what sort of tool they required and what was, and was not, acceptable for the final product. For example, artists may have ideas about specific interaction styles or the limits of processing time. The specification had to be carefully designed to ensure that time was not wasted researching unusable tools.

The smaller research projects (sections 5.3.1 and 5.3.2) provided the technical knowledge and understanding of production systems. Working on tools within the production pipeline adds constraints to research. On every project there have been added limitations relating to the software with which final tools must integrate. This meant that it has been necessary to understand in detail how the software works in order to know whether ideas will translate successfully into the production pipeline. Developing a knowledge of the nuances in each of the software interfaces added a significant overhead to the research projects. However, that overhead is removed from future projects once the knowledge is gained.

By contrast, using the production software can help to save time when developing tools. Using the software's programming interface can help greatly with providing functionality that would take a long time to implement oneself. For example, applications might have the ability to read image or geometry data, along with libraries of functions available specifically for working with content. This type of functionality would be very time-consuming to program. Therefore, when such functionality is required, development from within the software and using these libraries is much quicker than working on a standalone program.

Another benefit of developing software directly in the production pipeline is the ease of user testing. For example, during the rig removal project all development was completed within Nuke, therefore all testing could be done from within the Nuke interface. This helped in highlighting problems as they arose particularly if they were caused by intricacies of the Nuke software. However, an additional challenge was that fixing bugs in the code can be more difficult in production software than when developing a standalone application. For example, for some development work there was no debugger available that could be run alongside the code. Debugging had to be completed using manual methods which could be very time-consuming.

After working on the first project of rig removal some lessons were learnt that would help to speed up research on subsequent projects. For example, the set shift project was prototyped, which was much faster than the process used in the design stages of the segmentation project. The segmentation project had taught me the necessity of a really accurate tool so I focussed on testing this first before undertaking the integration issues. The set shift project also benefited from an existing knowledge of the Nuke API and what was possible within the structure of a Nuke plug-in.

Developing and implementing research in an industry environment has been insightful and has given me a practical understanding of generating useful content for a company like Aardman Animations. There are some challenges in combining the needs of academia with the needs of the company. For example, in an artistic-led company the size of Aardman Animations, there is only a small overlap between projects that they wish to pursue and ones that are academically acceptable. In addition to the limitation on projects, there is also the consideration of timescales. When searching for potential projects it was often found that projects the company wished to pursue were of a much shorter timescale than those favoured by the university. The work undertaken in this portfolio has highlighted some of the differing research requirements of this type of company in comparison to those of an academic institution.

## 5.8 Conclusions

Implementing research directly into industry is a valuable learning experience. It requires a researcher to gain an in depth knowledge of both state of the art research and industry methods and requirements. It is important to have a clear understanding of the context of the industry problem and a thorough knowledge of the company's processes as well as knowledge of state of the art academic research in that particular area. This chapter has discussed some of the additional challenges of integrating academic research into an industrial environment while satisfying both the industrial and university partners.

There are different methods of structuring the research, particularly the stage at which research is integrated with industry software. One issue highlighted by the research projects in this portfolio is the importance of deciding upon the best methods of developing research within industry. For example, the set shift project was structured very differently to the rig removal project.

A benefit of conducting research directly into production software is that a researcher immediately gets an understanding of the context of the problem. There is a clear end goal which provides the researcher with a specific brief and set of requirements to achieve. Additionally, working directly in the production pipeline means that any integration problems will arise during development, rather than at a later stage when trying to translate code into production software. Working directly in the production pipeline has the benefit of being able to easily complete testing and obtain user feedback at any stage of development. When developing directly on the software it is easier to demonstrate progress and test an artist's reaction to the tool. Although developing within the pipeline adds an initial overhead in learning the software environment, once this has been completed the development of subsequent tools is faster. In addition, the functionality offered by the development libraries can help to reduce development time considerably.

By contrast, completing research work directly in the production pipeline can also have a negative impact on the efficiency of the project. Developing directly within production software can increase overheads and implementation time. There tend to be added complications when working within software APIs, which can significantly increase the development time. This is particularly noticeable if the developer is unfamiliar with the environment, as it takes time to understand the software. The danger of this is that if the concept of the research is unsuccessful then additional time has been spent developing code that will not be used. The alternative is to develop the code in a separate environment and then test

a proof of concept before implementing within the software environment.

It was for these reasons that the tools developed in this research were completed in a variety of ways. It depends greatly on the the nature of the problem as to which approach is the most time effective. Generally each research project has been implemented in the way that allows for the fastest testing of the initial ideas. A result was found quickly during the set shift problem due to the fast development of a prototype. This meant that state of the art research could be tested quickly to see if a solution was possible. This approach led to a much quicker resolution than if each state of the art implementation had needed to be ported to a software plug-in for testing.

One of the main additional aspects which needs consideration when implementing research into industry are the integration and compatibility with the existing software in the company. It is important that these factors are considered at all stages of development and particularly during the design stages. Whether developing within the production software or in a prototype application, it is vital to have an in-depth knowledge of how the software API works and what functionalities it offers, so that solutions can be designed effectively. The overheads of working within the software decrease significantly once an understanding of developing tools for specific software has been gained. If knowledge of the capabilities of an API can be used when developing a solution in an alternative environment then it is less likely there will be problems when the solution is transferred to the specific software environment required.

Another notable overhead in the development of research for industrial application is the time required for testing implementations. Finalising a tool for end-users and providing support takes a lot of additional time after the development of novel research content. This extensive testing time is not considered for traditional academic research but is critical for industrial based research to be used in production.

The practice of implementing research work into industry software could be improved by an effort by production software companies to make integration easier. Some SDKs are much easier to use than others. For example, V-Ray is a relatively new piece of software and the interface has only a little documentation, which makes it more difficult for a new user to integrate research. By contrast, Maya has an extensive API with many built in libraries, which encourages the development of custom tools. If the SDKs make custom development more accessible, then it might be more common for research and development work to be integrated with that software. If software companies want to encourage customisation and research into new tools then an accessible and user-friendly API is important.

# Chapter 6

# Discussion and Conclusions

## 6.1 Discussion

This research has investigated the application of academic research work to the industrial process of stop-motion animation. It has looked specifically at how computational tools can aid the production process of stop-motion animation. Investigation into a number of different projects was requested over the duration of this research. Each research project highlighted a challenge currently faced in the production of stop-motion content and in which the development of technical tools could help to improve efficiency.

During the rig removal project, solutions to help with the segmentation of objects from video frames were investigated. This involved extensive research into state of the art techniques for image and video segmentation. After studying the benchmark capabilities in this area, it was necessary to investigate whether results could be improved to the level required by industry. I presented a novel video segmentation algorithm, which used the motion present in stop-motion frames to produce improved segmentation results. This delivered a notable contribution to knowledge in the area of improving image segmentation for rig removal and results showed improved accuracy when compared to a state of the art segmentation algorithm.

However, although a significant improvement in the segmentation of video content was made, I discovered that the company required results of a very high quality. As they wanted a production tool, the results needed to be both extremely accurate and faster than their current manual processes. The speed and accuracy required was of a very high level because results would be compared to results produced by very experienced and efficient users. The results from the improved segmentation tool in comparison to the final level of result required by Aardman Animations led me to conclude that a satisfactory production tool would not be achievable given the time and resources available. Sizeable advances on the state of the art in video segmentation would have been needed to build a tool capable of satisfying the requirements of post-production users. In addition to the level of improvement needed, the tools available for users had improved over the time that had been spent researching this project so the benchmark was higher than in the early stages of the investigation.

This realisation led to the need to focus research on a new area. Several potential projects were

investigated at this stage. The next project that was researched in depth was that of set shift, which is a problem that occurs on stop motion sets. The sets can shift over time and these movements have to be corrected in post-production. This project's research areas overlapped with some of those researched during the rig removal project, so it was important to ensure that the project avoided the known problem areas already discovered. For example, research into segmentation was avoided when researching set shift.

After significant research into tools for the correction of set shift, it was found that whilst it would be possible to develop a solution that could solve specific examples of set shift, it was not feasible to develop a solution that could cope with the range of shifts which occur. Many of the examples contained extremely subtle shifts and the tool would need to work to subpixel accuracy. From the results of studying the state of the art in this area, I concluded that the company would be best to continue with manual methods of correcting set shift until further advancements in the areas of edge detection, optical flow or segmentation occurred.

The next area of the production process to be investigated with the aim of advancing stop-motion tools, was the project of simulating plasticine computationally. This project was different to the previous projects in that an acceptable technical tool was already used. Out of the box methods were used in the company and, whilst that approach was acceptable, they requested to know if an improved shader could be generated. The results of the plasticine shading project showed that improvements could be made over the existing methods available in the production pipeline. After examining the plasticine material in detail using scientific methods, a new and novel shading model was developed. This new model improves the representation of plasticine computationally and fits more closely to the measured model of plasticine. This new model was then integrated into the production pipeline at Aardman Animations so that it would be available for modelling plasticine materials.

Conducting research over this range of projects means that several different aspects of the stop-motion animation production process have been assessed. To look at whether there is potential for computational tools to aid production, it is valuable to understand some of the challenges that occurred during the research. In conducting research within a non-research-led company and with the aim of producing academic research, several interesting conflicts arose. One of main difficulties in researching both for academia and industry was managing the timescales of both parties. At Aardman Animations previous research had been completed on a per-project basis and tended to be much shorter scale than purely academic research. Therefore, it was necessary to balance the timescale expectations of both parties and find compromises where necessary. Looking at the research into rig removal and set shift it was clear that although technical advancements could be made, they would not be to the standard required from a production tool. Becoming aware of the standard and quality required in production software was an important step in understanding whether stop-motion animation could be improved with technical tools and the resources available.

Another key challenge was the difficulty of finding research that could generate novel content and yet develop a production tool to the standard required by the company. Pertuzé et al. (2010) discuss the challenges associated with industry-university collaborations and define the Outcome-Impact Gap. The Outcome-Impact Gap describes the issue of generating interesting outcomes as well as industry impact. In this case, that is generating novel content for publication as well as useable tools in industry.

Therefore, in addition to matching timescales between parties, there was the secondary requirement to produce novel content and for investigations to be conducted with academic rigour. These were areas that a company such as Aardman Animations had not previously needed to consider in such detail. By contrast, the company's industry-based requirements focussed much more on quality of finish and usability.

The requirements of both timescale management and novel content meant that the selection of available projects was very restricted. At times it was difficult to find projects that satisfied both parties. This restriction was particularly prominent because I was working with a company that is strongly creatively-led and that does not have an existing research department or a research roadmap. This meant that there was no precedent of research projects to follow or any other long-term research in which to participate. These factors led to a distinction between completing research to generate novel content and developing technical advancements to deliver value to the production process. It has been a challenge to complete work that will generate novel content and be fully integrated into the production process in the time available.

As a result of satisfying the requirements of both parties, projects tended to end up with a very specific project brief. This well-defined brief may have contributed to the need to investigate several different projects. A drawback with working on such well-defined company-based projects is that it does mean the investigation is constrained. For example, some potential research areas may not be explored because of the limitations imposed by the company and the specific brief. However, industry-based research and purely-academic based research can compliment each other by working in these different manners, with the former being more constrained and the latter being more open to exploration.

I found that working in industry imposed a series of additional requirements on the research completed. The most prominent area of additional work in comparison to traditional academic research was the requirement for research to integrate into the existing production pipeline. Researching how the tool will be used in production and developing research for production users adds a significant overhead in terms of timescales and deliverables. It is necessary to learn how to develop for a particular piece of software and to gain understanding of how the applications work as a user. This is needed in order to understand the needs of the user and decide how the tool is best programmed. In addition to the time required to integrate the research into the production pipeline, a significant amount of time must be assigned to the iterative testing and fixing process required after rolling-out new tools. The ongoing testing and support is an overhead not required in purely academic research. In chapter 5, I discussed the benefits and challenges of working with research solutions directly in the production pipeline. It is very important in industry that consideration is given throughout the research process as to how the final tool would be used in production.

Working on academic research in a creative company without experience of long-term or academic research can be challenging. Conducting technical-based research in a creatively driven company with no dedicated research department meant that additional groundwork had to be completed in order to get the research projects and research infrastructure in place. The collaboration between university and industry had to be managed carefully in order to satisfy both parties with each research project. Understanding the requirements of both academic and industry partners provided a fascinating insight into the relationship between implementing academic research into a traditional, creatively-led industry.

## 6.2 Conclusions

This portfolio documents investigations into whether the implementation of technical research can improve the production process for stop-motion animation, without significantly affecting the traditional methods or changing the unique look and feel of the content. The project has led to research in a number of areas with various aspects of stop-motion animation being investigated.

The segmentation of video frames was researched with the aim of helping rig removal tasks. Rig removal is a post-production problem in which rigging objects that are captured in images on set must be removed from the footage. The research into a new video segmentation algorithm presented a novel contribution to research. This new algorithm uses both colour and motion information from the stop-motion footage and demonstrates improved results when compared to previous state of the art techniques.

Another post-production problem requested for investigation by Aardman Animations was that of set shift. The set shift project included research into areas such as edge detection and content warping. The aim was to investigate whether state of the art research could be used to detect and fix the subtle set shifts found in stop-motion footage. After investigation, it was concluded that this research would not be completable given the resources available and the complexity of the problem. The problem of set shift is an area where significant improvement in state of the art research is required before a production quality tool can be created.

The next project relating to stop-motion animation production was addressing the computational shading of plasticine materials. This project investigated the way that plasticine materials are simulated when created computationally. It explored whether current methods of simulating plasticine could be improved upon and studied the properties of plasticine using an experimental set-up. In this research it was discovered that improvements to the current shading model could be made. After analysis of the true physical properties of plasticine, a new shading model is presented which is shown to model plasticine more accurately than other state of the art solutions.

The tools created were done so within the production environment and integrated into the software used at Aardman Animations. They were tested for their ability to work within a real production environment and how they could successfully be utilised in an industry context.

In response to the original research question, I conclude that computational tools can definitely improve the production of stop motion animated content, as evidenced by the work completed in this portfolio. However, it is important to be careful about which areas are selected for research. As discovered by the first research project of rig removal, it is crucial to decide whether the desired end goal is likely to be achievable within the constraints of the project time and resources. With hindsight, I can see that the chances of improving on the state of the art for object segmentation were good, but the chances of improving segmentation to the level that was required by the company, in the time available, were much less likely. Therefore, in future I would complete a feasibility study at the beginning of an industry-university research collaboration project. It is vital to understand the requirements of the client in detail and what they are expecting to see in the end results.

The findings from the rig removal project helped in the decision to structure the second project of set shift differently. It was approached in a way that was more sensitive to the priorities of the company. It was clear from the experience researching rig removal that a tool was required that would be accurate

enough to avoid subsequent human intervention. If this was not possible, then from an industrial point of view the time would be better spent researching a different project. For this reason a faster method of prototyping was used to scope out whether the project had potential. I would recommend that this approach is considered for industry-university collaborations, as it allowed the exploration of an area but without spending additional time if the project was not feasible to the company.

Researching three different areas of stop-motion animation for Aardman Animations has produced a number of significant findings. Improvements and contributions have been made in both the areas of video segmentation and surface shading. Additionally research into set shift has answered the question set by the company about whether development of a tool in this area would be worth investing in at this stage. These contributions will help Aardman Animations in their understanding of these research topics and the tools available to help with these areas of the production process. There are many areas of future research work that Aardman Animations can investigate and it would be useful for them to remain informed about improvements in the state of the art research related to those discussed here. As new research is published they will now understand more clearly the quality of result they require in order for it to be useable in production. If the research is at the level required then it can be implemented as a plug-in and integrated into the production pipeline using the structure outlined by the projects completed in this portfolio.

Overall, this research has demonstrated that computational tools can help with improving the production of stop-motion animated content, with the dependency that it is important to choose carefully where that research time is directed. There are areas which can be improved with research but they are specific and research needs to be completed in a way that produces results to the standards required by industry. There are research areas which are not currently at a level useful to production and this was exemplified in the rig removal project. However, it is important that Aardman Animations keep current with areas of research such as this so that when new content in these research areas is released, they can be ready to use them to stream-line the production of stop-motion content.

Furthermore, it is important that traditional forms of animation, such as stop-motion animation, keep current with technical advancements and improving production methods. With a relatively low level of resources it is possible for Aardman Animations to keep up to date with advances in the state of the art research areas which are of importance to the progression of their tools. Animation is a very competitive business and there is constant pressure to keep costs to a minimum. If stop-motion animation is to stay competitive then it is important that it embraces new research and uses it to generate new tools which will help improve efficiency in the production pipeline.

# Bibliography


Adobe. Adobe After Effects, 2012. URL http://www.adobe.com/AfterEffects. [Online; accessed 12-May-2012].

A. Agarwala, M. Dontcheva, M. Agrawala, S. M. Drucker, A. Colburn, B. Curless, D. Salesin, and M. F. Cohen. Interactive digital photomontage. *ACM Trans. Graph.*, 23(3):294–302, 2004a. doi: http://doi.acm.org/10.1145/1015706.1015718.

A. Agarwala, A. Hertzmann, D. Salesin, and S. M. Seitz. Keyframe-based tracking for rotoscoping and animation. *ACM Trans. Graph.*, 23(3):584–591, 2004b. doi: http://doi.acm.org/10.1145/1015706.1015764.

P. F. Alcantarilla, A. Bartoli, and A. J. Davison. KAZE features. In A. W. Fitzgibbon, S. Lazebnik, P. Perona, Y. Sato, and C. Schmid, editors, *Computer Vision - ECCV 2012 - 12th European Conference on Computer Vision, Florence, Italy, October 7-13, 2012, Proceedings, Part VI*, volume 7577 of *Lecture Notes in Computer Science*, pages 214–227. Springer, 2012. ISBN 978-3-642-33782-6. doi: http://dx.doi.org/10.1007/978-3-642-33783-3_16.

M. A. Ali, I. Sato, T. Okabe, and Y. Sato. Toward efficient acquisition of BRDFs with fewer samples. In K. M. Lee, Y. Matsushita, J. M. Rehg, and Z. Hu, editors, *Computer Vision - ACCV 2012, 11th Asian Conference on Computer Vision, Daejeon, Korea, November 5-9, 2012, Revised Selected Papers, Part IV*, volume 7727 of *Lecture Notes in Computer Science*, pages 54–67. Springer, 2012. ISBN 978-3-642-37447-0. doi: http://dx.doi.org/10.1007/978-3-642-37447-0_5.

P. Arbelaez, M. Maire, C. C. Fowlkes, and J. Malik. From contours to regions: An empirical evaluation. In *2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2009), 20-25 June 2009, Miami, Florida, USA*, pages 2294–2301. IEEE, 2009. ISBN 978-1-4244-3992-8. doi: http://doi.ieeecomputersociety.org/10.1109/CVPRW.2009.5206707.

P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik. Contour detection and hierarchical image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 33(5):898–916, 2011. doi: http://dx.doi.org/10.1109/TPAMI.2010.161.

M. Ashikhmin and S. Premoze. Distribution-based BRDFs. Technical report, University of Utah, 2007.

M. Ashikhmin and P. Shirley. An anisotropic phong BRDF model. *J. Graphics Tools*, 5(2):25–32, 2000. doi: http://dx.doi.org/10.1080/10867651.2000.10487522.

Autodesk. Flame, 2014. URL http://www.autodesk.co.uk/products/flame-family/overview. [Online; accessed 13-April-2014].

V. Badrinarayanan, I. Budvytis, and R. Cipolla. Semi-supervised video segmentation using tree structured graphical models. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(11):2751–2764, 2013a. doi: http://doi.ieeecomputersociety.org/10.1109/TPAMI.2013.54.

V. Badrinarayanan, I. Budvytis, and R. Cipolla. Mixture of trees probabilistic graphical model for video segmentation. pages 1–16. Springer, 2013b.

J. Bai, A. Agarwala, M. Agrawala, and R. Ramamoorthi. Selectively de-animating video. *ACM Trans. Graph.*, 31(4):66, 2012. doi: http://doi.acm.org/10.1145/2185520.2185562.

X. Bai and G. Sapiro. A geodesic framework for fast interactive image and video segmentation and matting. In *IEEE 11th International Conference on Computer Vision, ICCV 2007, Rio de Janeiro, Brazil, October 14-20, 2007*, pages 1–8. IEEE, 2007. doi: http://dx.doi.org/10.1109/ICCV.2007.4408931.

X. Bai, J. Wang, D. Simons, and G. Sapiro. Video snapcut: Robust video object cutout using localized classifiers. In *ACM SIGGRAPH 2009 Papers*, SIGGRAPH '09, pages 70:1–70:11, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-726-4. doi: 10.1145/1576246.1531376. URL http://doi.acm.org/10.1145/1576246.1531376.

X. Bai, J. Wang, and G. Sapiro. Dynamic color flow: A motion-adaptive color model for object segmentation in video. In K. Daniilidis, P. Maragos, and N. Paragios, editors, *Computer Vision - ECCV 2010 - 11th European Conference on Computer Vision, Heraklion, Crete, Greece, September 5-11, 2010, Proceedings, Part V*, volume 6315 of *Lecture Notes in Computer Science*, pages 617–630. Springer, 2010. ISBN 978-3-642-15554-3. doi: http://dx.doi.org/10.1007/978-3-642-15555-0_45.

H. Bay, A. Ess, T. Tuytelaars, and L. J. V. Gool. Speeded-up robust features (SURF). *Computer Vision and Image Understanding*, 110(3):346–359, 2008. doi: http://dx.doi.org/10.1016/j.cviu.2007.09.014.

T. Beier and S. Neely. Feature-based image metamorphosis. In J. J. Thomas, editor, *Proceedings of the 19st Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 1992*, pages 35–42. ACM, 1992. doi: http://doi.acm.org/10.1145/133994.134003.

J. Biggins. private communication, 2014. Post-Production Artist, Aardman Animations.

J. F. Blinn. Models of light reflection for computer synthesized pictures. *SIGGRAPH Comput. Graph.*, 11(2):192–198, July 1977. ISSN 0097-8930. doi: 10.1145/965141.563893.

G. Borgefors. Distance transformations in digital images. *Computer Vision, Graphics, and Image Processing*, 34(3):344–371, 1986. doi: http://dx.doi.org/10.1016/S0734-189X(86)80047-0.

Y. Boykov and M.-P. Jolly. Interactive graph cuts for optimal boundary amp; region segmentation of objects in n-d images. In *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, volume 1, pages 105–112 vol.1, 2001. doi: 10.1109/ICCV.2001.937505.

T. Brox and J. Malik. Object segmentation by long term analysis of point trajectories. In *Proceedings of the 11th European Conference on Computer Vision: Part V*, ECCV'10, pages 282–295, Berlin, Heidelberg, 2010. Springer-Verlag. ISBN 3-642-15554-5, 978-3-642-15554-3. URL http://dl.acm.org/citation.cfm?id=1888150.1888173.

B. Burley. Physically-based shading at disney, 2012. URL http://disney-animation.s3.amazonaws.com/library/s2012_pbs_disney_brdf_notes_v2.pdf. [Online; accessed 11-March-2014].

J. Canny. A computational approach to edge detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 8(6): 679–698, 1986. doi: http://dx.doi.org/10.1109/TPAMI.1986.4767851.

C.-H. Chang, Y. Sato, and Y.-Y. Chuang. Shape-preserving half-projective warps for image stitching. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 3254–3261, June 2014. doi: 10.1109/CVPR.2014.422.

Chaos Group. V-Ray for Maya, 2014. URL http://www.chaosgroup.com/en/2/vray_maya.html. [Online; accessed 15-January-2014].

H.-Y. Chen, Y.-Y. Lin, and B.-Y. Chen. Robust feature matching with alternate hough and inverted hough transforms. In *2013 IEEE Conference on Computer Vision and Pattern Recognition, Portland, OR, USA, June 23-28, 2013*, pages 2762–2769. IEEE, 2013. doi: http://dx.doi.org/10.1109/CVPR.2013.356.

R. L. Cook and K. E. Torrance. A reflectance model for computer graphics. *ACM Trans. Graph.*, 1(1): 7–24, 1982. doi: http://doi.acm.org/10.1145/357290.357293.

D. Corrigan, S. Robinson, and A. Kokaram. Video matting using motion extended grabcut. In *Visual Media Production (CVMP 2008), 5th European Conference on*, pages 1–9, Nov 2008.

A. Criminisi, T. Sharp, and A. Blake. Geos: Geodesic image segmentation. In D. A. Forsyth, P. H. S. Torr, and A. Zisserman, editors, *Computer Vision - ECCV 2008, 10th European Conference on Computer Vision, Marseille, France, October 12-18, 2008, Proceedings, Part I*, volume 5302 of *Lecture Notes in Computer Science*, pages 99–112. Springer, 2008. ISBN 978-3-540-88681-5. doi: http://dx.doi.org/10.1007/978-3-540-88682-2_9.

A. Criminisi, T. Sharp, C. Rother, and P. Pérez. Geodesic image and video editing. *ACM Trans. Graph.*, 29(5):134, 2010. doi: http://doi.acm.org/10.1145/1857907.1857910.

Disney. BRDF explorer, 2012. URL http://www.disneyanimation.com/technology/brdf.html. [Online; accessed 17-April-2014].

P. Dollár and C. L. Zitnick. Structured forests for fast edge detection. In *IEEE International Conference on Computer Vision, ICCV 2013, Sydney, Australia, December 1-8, 2013*, pages 1841–1848. IEEE, 2013. doi: http://dx.doi.org/10.1109/ICCV.2013.231.

R. Dondera, V. I. Morariu, Y. Wang, and L. S. Davis. Interactive video segmentation using occlusion boundaries and temporally coherent superpixels. In *IEEE Winter Conference on Applications of*

*Computer Vision, Steamboat Springs, CO, USA, March 24-26, 2014*, pages 784–791. IEEE, 2014. doi: http://dx.doi.org/10.1109/WACV.2014.6836023.

C. Donner, T. Weyrich, E. d'Eon, R. Ramamoorthi, and S. Rusinkiewicz. A layered, heterogeneous reflectance model for acquiring and rendering human skin. *ACM Trans. Graph.*, 27(5):140:1–140:12, Dec. 2008. ISSN 0730-0301. doi: 10.1145/1409060.1409093. URL http://doi.acm.org/10.1145/1409060.1409093.

J. Dorsey, H. Rushmeier, and F. Sillion. Advanced material appearance modeling. In *ACM SIGGRAPH 2008 Classes*, SIGGRAPH '08, pages 5:1–5:145, New York, NY, USA, 2008. ACM. doi: 10.1145/1401132.1401140. URL http://doi.acm.org/10.1145/1401132.1401140.

I. Failes. Band of Misfits: Dnegs pirate adventures, 2012. URL http://www.fxguide.com/featured/band-of-misfits-dnegs-pirate-adventures/.

J. Filip, R. Vvra, M. Haindl, P. Zid, M. Krupika, and V. Havran. BRDF slices: Accurate adaptive anisotropic appearance acquisition. In *2013 IEEE Conference on Computer Vision and Pattern Recognition, Portland, OR, USA, June 23-28, 2013*, pages 1468–1473. IEEE, 2013. doi: http://dx.doi.org/10.1109/CVPR.2013.193.

S. C. Foo. A gonioreflectometer for measuring the bidirectional reflectance of material for use in illumination computation. Master's thesis, Cornell University, Ithaca, New York, July 1997. URL http://www.graphics.cornell.edu/research/measure/sfoo-thesis.pdf.

F. Galasso, R. Cipolla, and B. Schiele. Video segmentation with superpixels. In K. M. Lee, Y. Matsushita, J. M. Rehg, and Z. Hu, editors, *Computer Vision - ACCV 2012 - 11th Asian Conference on Computer Vision, Daejeon, Korea, November 5-9, 2012, Revised Selected Papers, Part I*, volume 7724 of *Lecture Notes in Computer Science*, pages 760–774. Springer, 2012. ISBN 978-3-642-37330-5. doi: http://dx.doi.org/10.1007/978-3-642-37331-2_57.

F. Galasso, N. S. Nagaraja, T. J. Cardenas, T. Brox, and B. Schiele. A unified video segmentation benchmark: Annotation, metrics and analysis. In *IEEE International Conference on Computer Vision, ICCV 2013, Sydney, Australia, December 1-8, 2013*, pages 3527–3534. IEEE, 2013. doi: http://dx.doi.org/10.1109/ICCV.2013.438.

A. Ghosh, W. Heidrich, S. Achutha, and M. O. Toole. A basis illumination approach to BRDF measurement. *International Journal of Computer Vision*, 90(2):183–197, 2010. doi: http://dx.doi.org/10.1007/s11263-008-0151-7.

M. Gleicher. Image snapping. In *Proceedings of the 22Nd Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '95, pages 183–190, New York, NY, USA, 1995. ACM. ISBN 0-89791-701-4. doi: 10.1145/218380.218441. URL http://doi.acm.org/10.1145/218380.218441.

M. Glencross, G. J. Ward, F. Melendez, C. Jay, J. Liu, and R. Hubbold. A perceptually validated model for surface depth hallucination. In *ACM SIGGRAPH 2008 Papers*, SIGGRAPH '08, pages 59:1–59:8,

New York, NY, USA, 2008. ACM. ISBN 978-1-4503-0112-1. doi: 10.1145/1399504.1360658. URL http://doi.acm.org/10.1145/1399504.1360658.

Guinness World Records. First animated film, 2014. URL \url{http://www.guinnessworldrecords.com/records-3000/first-animated-film/}.

J. Günther, T. Chen, M. Goesele, I. Wald, and H.-P. Seidel. Efficient acquisition and realistic rendering of car paint. In G. Greiner, J. Hornegger, H. Niemann, and M. Stamminger, editors, *Proceedings of 10th International Fall Workshop - Vision, Modeling, and Visualization (VMV) 2005*, pages 487–494. Akademische Verlagsgesellschaft Aka GmbH, Nov. 2005. ISBN 3-89838-068-8.

C. Harris and M. Stephens. A combined corner and edge detector. In C. J. Taylor, editor, *Proceedings of the Alvey Vision Conference, AVC 1988, Manchester, UK, September, 1988*, pages 1–6. Alvey Vision Club, 1988. doi: http://dx.doi.org/10.5244/C.2.23.

X. D. He, K. E. Torrance, F. X. Sillion, and D. P. Greenberg. A comprehensive physical model for light reflection. In J. J. Thomas, editor, *Proceedings of the 18st Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 1991*, pages 175–186. ACM, 1991. ISBN 0-89791-436-8. doi: http://doi.acm.org/10.1145/122718.122738.

N. Hoffman. Background: Physics and math of shading, 2012. URL http://blog.selfshadow.com/publications/s2012-shading-course/hoffman/s2012_pbs_physics_math_notes.pdf. [Online; accessed 04-April-2014].

M. Isard and A. Blake. CONDENSATION - conditional density propagation for visual tracking. *International Journal of Computer Vision*, 29(1):5–28, 1998. doi: http://dx.doi.org/10.1023/A:1008078328650.

A. Jacobson, I. Baran, J. Popovic, and O. Sorkine. Bounded biharmonic weights for real-time deformation. *ACM Trans. Graph.*, 30(4):78, 2011. doi: http://doi.acm.org/10.1145/2010324.1964973.

Jaloxa. WebHDR camera calibration, 2011. URL http://www.jaloxa.eu/webhdr/calibrate.shtml. [Online; accessed 04-March-2014].

K. Jensen and D. Anastassiou. Subpixel edge localization and the interpolation of still images. *IEEE Transactions on Image Processing*, 4(3):285–295, 1995. doi: http://dx.doi.org/10.1109/83.366477.

R. E. Kalman. A new approach to linear filtering and prediction problems. *Journal of basic Engineering*, 82(1):35–45, 1960.

J. Kim, R. Horstmeyer, I.-J. Kim, and R. Raskar. Highlighted depth-of-field photography: Shining light on focus. *ACM Trans. Graph.*, 30(3):24, 2011. doi: http://doi.acm.org/10.1145/1966394.1966403.

P. Kohli and P. H. S. Torr. Dynamic graph cuts for efficient inference in markov random fields. *IEEE Trans. Pattern Anal. Mach. Intell.*, 29(12):2079–2088, 2007. doi: http://doi.ieeecomputersociety.org/10.1109/TPAMI.2007.1128.

A. Kokaram, B. Collis, and S. Robinson. Practical motion based video matting. In *proceedings of the IEE European Conference on Visual Media Production (CVMP'05)*, pages 130–136, 2005a.

A. C. Kokaram, B. Collis, and S. Robinson. Automated rig removal with Bayesian motion interpolation. *IEE Proceedings of Vision, Image and Signal Processing*, 152:407–414, Aug. 2005b.

E. P. F. Lafortune, S.-C. Foo, K. E. Torrance, and D. P. Greenberg. Non-linear approximation of reflectance functions. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '97, pages 117–126, New York, NY, USA, 1997. ACM Press/Addison-Wesley Publishing Co. ISBN 0-89791-896-7. doi: 10.1145/258734.258801. URL http://dx.doi.org/10.1145/258734.258801.

S. Leutenegger, M. Chli, and R. Siegwart. BRISK: Binary robust invariant scalable keypoints. In D. N. Metaxas, L. Quan, A. Sanfeliu, and L. J. V. Gool, editors, *IEEE International Conference on Computer Vision, ICCV 2011, Barcelona, Spain, November 6-13, 2011*, pages 2548–2555. IEEE, 2011. ISBN 978-1-4577-1101-5. doi: http://dx.doi.org/10.1109/ICCV.2011.6126542.

A. Levin, D. Lischinski, and Y. Weiss. A closed form solution to natural image matting. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2006), 17-22 June 2006, New York, NY, USA*, pages 61–68. IEEE Computer Society, 2006. ISBN 0-7695-2597-0. doi: http://doi.ieeecomputersociety.org/10.1109/CVPR.2006.18.

H. Li and C. Shen. Interactive color image segmentation with linear programming. *Mach. Vis. Appl.*, 21(4):403–412, 2010. doi: http://springerlink.metapress.com/content/b254775776114226/.

Y. Li, J. Sun, C.-K. Tang, and H.-Y. Shum. Lazy snapping. *ACM Trans. Graph.*, 23(3):303–308, 2004. doi: http://doi.acm.org/10.1145/1015706.1015719.

F. Liu, M. Gleicher, H. Jin, and A. Agarwala. Content-preserving warps for 3d video stabilization. In *ACM SIGGRAPH 2009 Papers*, SIGGRAPH '09, pages 44:1–44:9, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-726-4. doi: 10.1145/1576246.1531350. URL http://doi.acm.org/10.1145/1576246.1531350.

D. G. Lowe. Object recognition from local scale-invariant features. In *ICCV*, pages 1150–1157, 1999. doi: http://computer.org/proceedings/iccv/0164/vol202/01641150abs.htm.

D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004. doi: http://dx.doi.org/10.1023/B:VISI.0000029664.99615.94.

B. D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In P. J. Hayes, editor, *Proceedings of the 7th International Joint Conference on Artificial Intelligence (IJCAI 81), Vancouver, BC, Canada, August 1981*, pages 674–679. William Kaufmann, 1981.

E. P. Lyvers, O. R. Mitchell, M. L. Akey, and A. P. Reeves. Subpixel measurements using a moment-based edge operator. *IEEE Trans. Pattern Anal. Mach. Intell.*, 11(12):1293–1309, 1989. doi: http://computer.org/tpami/tp1989/i1293abs.htm.

D. Marr and E. Hildreth. Theory of edge detection. *Proceedings of the Royal Society of London. Series B. Biological Sciences*, 207(1167):187–217, 1980.

S. R. Marschner, S. H. Westin, E. P. Lafortune, K. E. Torrance, and D. P. Greenberg. Image-based BRDF measurement including human skin. In D. Lischinski and G. W. Larson, editors, *Rendering Techniques 99, Proceedings of the Eurographics Workshop in Granada, Spain, June 21-23, 1999*, pages 131–144. Springer, 1999. ISBN 3-211-83382-X.

S. R. Marschner, H. W. Jensen, M. Cammarano, S. Worley, and P. Hanrahan. Light scattering from human hair fibers. *ACM Trans. Graph.*, 22(3):780–791, 2003. doi: http://doi.acm.org/10.1145/882262.882345.

M. Marszatek and C. Schmid. Accurate object localization with shape masks. In *Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on*, pages 1–8, June 2007. doi: 10.1109/CVPR.2007.383085.

D. R. Martin, C. Fowlkes, and J. Malik. Learning to detect natural image boundaries using local brightness, color, and texture cues. *IEEE Trans. Pattern Anal. Mach. Intell.*, 26(5):530–549, 2004. doi: http://csdl.computer.org/comp/trans/tp/2004/05/i0530abs.htm.

W. Matusik. *A Data-Driven Reflectance Model*. PhD thesis, Massachusetts Institute of Technology, 2003. URL http://people.csail.mit.edu/wojciech/pubs/phdthesis.pdf.

W. Matusik, H. Pfister, M. Brand, and L. McMillan. A data-driven reflectance model. *ACM Transactions on Graphics*, 22(3):759–769, July 2003.

MERL. MERL BRDF database, 2006. URL http://www.merl.com/brdf/. [Online; accessed 16-March-2014].

H. Moravec. Obstacle avoidance and navigation in the real world by a seeing robot rover. In *tech. report CMU-RI-TR-80-03, Robotics Institute, Carnegie Mellon University & doctoral dissertation, Stanford University*, number CMU-RI-TR-80-03. September 1980.

E. N. Mortensen and W. A. Barrett. Intelligent scissors for image composition. In *Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '95, pages 191–198, New York, NY, USA, 1995. ACM. ISBN 0-89791-701-4. doi: 10.1145/218380.218442. URL http://doi.acm.org/10.1145/218380.218442.

S. Narasimhan. Vision sensors: Radiometry and radiometric calibration, 2008. URL http://www.cs.cmu.edu/~ILIM/courses/vision-sensors/lectures/radiometric-calibration.ppt. [Online; accessed 03-March-2014].

A. Ngan, F. Durand, and W. Matusik. Experimental analysis of BRDF models. In O. Deussen, A. Keller, K. Bala, P. Dutr, D. W. Fellner, and S. N. Spencer, editors, *Proceedings of the Eurographics Symposium on Rendering Techniques, Konstanz, Germany, June 29 - July 1, 2005*, pages 117–126. Eurographics Association, 2005. ISBN 3-905673-23-1. doi: http://www.eg.org/EG/DL/WS/EGWR/EGSR05/117-126.pdf.

P. Ochs, J. Malik, and T. Brox. Segmentation of moving objects by long term video analysis. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 36(6):1187–1200, June 2014. ISSN 0162-8828. doi: 10.1109/TPAMI.2013.242.

M. Oren and S. K. Nayar. Generalization of Lambert's reflectance model. In *Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '94, pages 239–246, New York, NY, USA, 1994. ACM. ISBN 0-89791-667-0. doi: 10.1145/192161.192213. URL http://doi.acm.org/10.1145/192161.192213.

A. Papazoglou and V. Ferrari. Fast object segmentation in unconstrained video. In *IEEE International Conference on Computer Vision, ICCV 2013, Sydney, Australia, December 1-8, 2013*, pages 1777–1784. IEEE, 2013. doi: http://dx.doi.org/10.1109/ICCV.2013.223.

S. Paris and F. Durand. A topological approach to hierarchical segmentation using mean shift. In *Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on*, pages 1–8, June 2007. doi: 10.1109/CVPR.2007.383228.

D. Park, C. L. Zitnick, D. Ramanan, and P. Dollr. Exploring weak stabilization for motion feature extraction. In *2013 IEEE Conference on Computer Vision and Pattern Recognition, Portland, OR, USA, June 23-28, 2013*, pages 2882–2889. IEEE, 2013. doi: http://dx.doi.org/10.1109/CVPR.2013.371.

K. Perlin. Improving noise. In T. Appolloni, editor, *Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 2002, San Antonio, Texas, USA, July 23-26, 2002*, pages 681–682. ACM, 2002. ISBN 1-58113-521-1. doi: http://doi.acm.org/10.1145/566570.566636.

J. A. Pertuzé, E. S. Calder, E. M. Greitzer, and W. A. Lucas. Best Practices for Industry-University Collaboration. *MIT Sloan Management Review*, 51(4):83–90, 2010.

B. T. Phong. Illumination for computer generated pictures. *Commun. ACM*, 18(6):311–317, June 1975. ISSN 0001-0782. doi: 10.1145/360825.360839. URL http://doi.acm.org/10.1145/360825.360839.

J. M. Prewitt. Object enhancement and extraction. *Picture processing and Psychopictorics*, 10(1): 15–19, 1970.

B. L. Price, B. S. Morse, and S. Cohen. LIVEcut: Learning-based interactive video segmentation by evaluation of multiple propagated cues. In *IEEE 12th International Conference on Computer Vision, ICCV 2009, Kyoto, Japan, September 27 - October 4, 2009*, pages 779–786. IEEE, 2009. doi: http://dx.doi.org/10.1109/ICCV.2009.5459293.

L. G. Roberts. *Machine Perception of Three-Dimensional Solids*. Outstanding Dissertations in the Computer Sciences. Garland Publishing, New York, 1963. ISBN 0-8240-4427-4.

M. A. Robertson, S. Borman, and R. L. Stevenson. Estimation-theoretic approach to dynamic range enhancement using multiple exposures. *J. Electronic Imaging*, 12(2):219–228, 2003. doi: http://dx.doi.org/10.1117/1.1557695.

C. Rother, V. Kolmogorov, and A. Blake. Grabcut : interactive foreground extraction using iterated graph cuts. *ACM Trans. Graph.*, 23(3):309–314, 2004. doi: http://doi.acm.org/10.1145/1015706.1015720.

J. Rüegg, O. Wang, A. Smolic, and M. H. Gross. DuctTake: Spatiotemporal video compositing. *Comput. Graph. Forum*, 32(2):51–61, 2013. doi: http://dx.doi.org/10.1111/cgf.12025.

M. Rump, G. Müller, R. Sarlette, D. Koch, and R. Klein. Photo-realistic rendering of metallic car paint from image-based measurements. *Computer Graphics Forum*, 27(2):527–536, Apr. 2008.

S. Rusinkiewicz. A new change of variables for efficient BRDF representation. In G. Drettakis and N. L. Max, editors, *Rendering Techniques 98, Proceedings of the Eurographics Workshop in Vienna, Austria, June 29 - July 1, 1998*, pages 11–22. Springer, 1998. ISBN 3-211-83213-0.

I. Sadeghi, H. Pritchett, H. W. Jensen, and R. Tamstorf. An artist friendly hair shading system. In *ACM SIGGRAPH 2010 Papers*, SIGGRAPH '10, pages 56:1–56:10, New York, NY, USA, 2010. ACM. ISBN 978-1-4503-0210-4. doi: 10.1145/1833349.1778793. URL http://doi.acm.org/10.1145/1833349.1778793.

I. Sadeghi, O. Bisker, J. D. Deken, and H. W. Jensen. A practical microcylinder appearance model for cloth rendering. *ACM Trans. Graph.*, 32(2):14, 2013. doi: http://doi.acm.org/10.1145/2451236.2451240.

C. Schlick. An inexpensive BRDF model for physically-based rendering. *Comput. Graph. Forum*, 13 (3):233–246, 1994.

C. Schmid and R. Mohr. Local grayvalue invariants for image retrieval. *IEEE Trans. Pattern Anal. Mach. Intell.*, 19(5):530–535, 1997. doi: http://www.computer.org/tpami/tp1997/i0530abs.htm.

T. Schwarz. BRDF acquisition device, 2012. URL http://www.tomschwarz.co.uk/design/brdf-acquisition-device/. [Online; accessed 03-May-2014].

M. Seymour. The art of roto: 2011, Oct. 2011. URL http://www.fxguide.com/featured/the-art-of-roto-2011/.

J. Shotton, A. Blake, and R. Cipolla. Contour-based learning for object detection. In *10th IEEE International Conference on Computer Vision (ICCV 2005), 17-20 October 2005, Beijing, China*, pages 503–510. IEEE Computer Society, 2005. ISBN 0-7695-2334-X. doi: http://doi.ieeecomputersociety.org/10.1109/ICCV.2005.63.

J. Sivic, B. C. Russell, A. A. Efros, A. Zisserman, and W. T. Freeman. Discovering objects and their localization in images. In *10th IEEE International Conference on Computer Vision (ICCV 2005), 17-20 October 2005, Beijing, China*, pages 370–377. IEEE Computer Society, 2005. ISBN 0-7695-2334-X. doi: http://doi.ieeecomputersociety.org/10.1109/ICCV.2005.77.

B. G. Smith. Geometrical shadowing of a random rough surface. *IEEE Trans. on Antennas and Propagation*, pages 668–671, september 1967. doi: 10.1109/TAP.1967.1138991.

I. Sobel and G. Feldman. A 3x3 isotropic gradient operator for image processing. 1968. Presented at a talk at the Stanford Artificial Project. Available at http://www.researchgate.net/publication/239398674_An_Isotropic_3_3_Image_Gradient_Operator.

R. Spence. private communication, 2014. CG Supervisor, Aardman Animations.

M. M. Stark, J. Arvo, and B. E. Smits. Barycentric parameterizations for isotropic BRDFs. *IEEE Trans. Vis. Comput. Graph.*, 11(2):126–138, 2005. doi: http://dx.doi.org/10.1109/TVCG.2005.26.

J. Sun, J. Sun, S. B. Kang, Z.-B. Xu, X. Tang, and H.-Y. Shum. Flash cut: Foreground extraction with flash and no-flash image pairs. In *Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on*, pages 1–8, June 2007. doi: 10.1109/CVPR.2007.383080.

P. Sundberg, T. Brox, M. Maire, P. Arbelaez, and J. Malik. Occlusion boundary detection and figure/ground assignment from optical flow. In *Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '11, pages 2233–2240, Washington, DC, USA, 2011. IEEE Computer Society. ISBN 978-1-4577-0394-2. doi: 10.1109/CVPR.2011.5995364. URL http://dx.doi.org/10.1109/CVPR.2011.5995364.

Supercrumbly. abTwoFace, Sept. 2004. URL \url{http://www.supercrumbly.com/3d.php?sid=113\#.U5q5rkkX-Z9}. [Online; accessed 12-November-2012].

A. J. Tabatabai and O. R. Mitchell. Edge location to subpixel values in digital imagery. *IEEE Trans. Pattern Anal. Mach. Intell.*, 6(2):188–201, 1984. doi: http://dx.doi.org/10.1109/TPAMI.1984.4767502.

J. Talbot. Coursework - implementing grabcut, May 2010. URL http://www.justintalbot.com/course-work/.

The Foundry. FURNACECORE for NUKE — VectorGenerator, 2013a. URL http://www.thefoundry.co.uk/articles/2010/05/05/117/furnacecore-for-nuke-vectorgenerator/. [Online; accessed 17-March-2013].

The Foundry. NUKE and NUKEX — Advancing the art of digital compositing, 2013b. URL http://www.thefoundry.co.uk/products/nuke/.

The Foundry. Nuke: Latest release, 2014a. URL \url{http://www.thefoundry.co.uk/products/nuke-product-family/nuke/latest-version/}. [Online; accessed 12-May-2014].

The Foundry. Nuke NDK developer guide, 2014b. URL \url{http://docs.thefoundry.co.uk/nuke/80/ndkdevguide/}. [Online; accessed 19-January-2012].

The Pixel Farm. PFTrack, 2014. URL http://www.thepixelfarm.co.uk/products/PFTrack. [Online; accessed 16-June-2014].

P. J. Toivanen. New geodesic distance transforms for gray-scale images. *Pattern Recogn. Lett.*, 17(5):437–450, May 1996. ISSN 0167-8655. doi: 10.1016/0167-8655(96)00010-4. URL http://dx.doi.org/10.1016/0167-8655(96)00010-4.

C. Tomasi and T. Kanade. Detection and tracking of point features. Technical report, International Journal of Computer Vision, 1991.

K. E. Torrance and E. M. Sparrow. Radiometry. chapter Theory for Off-specular Reflection from Roughened Surfaces, pages 32–41. Jones and Bartlett Publishers, Inc., USA, 1992. ISBN 0-86720-294-7. URL http://dl.acm.org/citation.cfm?id=136913.136924.

C. Trippe. Pattern generation, 2009. URL http://www.ctvisualeffects.com/renderman/maya_pattern_animation/pattern_animation.html. [Online; accessed 12-November-2013].

VisionBib. Subpixel accuracy edge detection methods, sub-pixel edges, 2014. URL http://www.visionbib.com/bibliography/edge227.html. [Online; accessed 17-June-2013].

L. von Ahn, R. Liu, and M. Blum. Peekaboom: a game for locating objects in images. In *Proceedings of the SIGCHI conference on Human Factors in computing systems*, CHI '06, pages 55–64, New York, NY, USA, 2006. ACM. ISBN 1-59593-372-7.

B. Walter, S. R. Marschner, H. Li, and K. E. Torrance. Microfacet models for refraction through rough surfaces. In J. Kautz and S. N. Pattanaik, editors, *Proceedings of the Eurographics Symposium on Rendering Techniques, Grenoble, France, 2007*, pages 195–206. Eurographics Association, 2007. ISBN 978-3-905673-52-4. doi: http://dx.doi.org/10.2312/EGWR/EGSR07/195-206.

J. Wang, P. Bhat, A. Colburn, M. Agrawala, and M. F. Cohen. Interactive video cutout. *ACM Trans. Graph.*, 24(3):585–594, 2005. doi: http://doi.acm.org/10.1145/1073204.1073233.

J. Wang, S. Zhao, X. Tong, J. Snyder, and B. Guo. Modeling anisotropic surface reflectance with example-based microfacet synthesis. In *ACM SIGGRAPH 2008 Papers*, SIGGRAPH '08, pages 41:1–41:9, New York, NY, USA, 2008. ACM. ISBN 978-1-4503-0112-1. doi: 10.1145/1399504.1360640. URL http://doi.acm.org/10.1145/1399504.1360640.

T. Wang and J. P. Collomosse. Probabilistic motion diffusion of labeling priors for coherent video segmentation. *IEEE Transactions on Multimedia*, 14(2):389–400, 2012. doi: http://dx.doi.org/10.1109/TMM.2011.2177078.

T. Wang, B. Han, and J. P. Collomosse. Touchcut: Fast image and video segmentation using single-touch interaction. *Computer Vision and Image Understanding*, 120:14–30, 2014. doi: http://dx.doi.org/10.1016/j.cviu.2013.10.013.

G. J. Ward. Measuring and modeling anisotropic reflection. In J. J. Thomas, editor, *Proceedings of the 19st Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 1992*, pages 265–272. ACM, 1992. doi: http://doi.acm.org/10.1145/133994.134078.

S. Weintraub. 15 things to know about THE PIRATES! BAND OF MISFITS from our set visit; plus an interview with director Peter Lord, 2012. URL \url{http://collider.com/peter-lord-pirates-band-of-misfits-interview/}.

N. Widynski and M. Mignotte. A particle filter framework for contour detection. In A. W. Fitzgibbon, S. Lazebnik, P. Perona, Y. Sato, and C. Schmid, editors, *Computer Vision - ECCV 2012 - 12th European Conference on Computer Vision, Florence, Italy, October 7-13, 2012, Proceedings, Part I*, volume 7572 of *Lecture Notes in Computer Science*, pages 780–793. Springer, 2012. ISBN 978-3-642-33717-8. doi: http://dx.doi.org/10.1007/978-3-642-33718-5_56.

G. Wolberg. Recent advances in image morphing. In *Computer Graphics International, 1996. Proceedings*, pages 64–71. IEEE, 1996.

Q. Ying-Dong, C. Cheng-Song, C. San-Ben, and L. Jin-Quan. A fast subpixel edge detection method using Sobel-Zernike moments operator. *Image Vision Comput.*, 23(1):11–17, 2005. doi: http://dx.doi.org/10.1016/j.imavis.2004.07.003.

Z. Zhou, H. Jin, and Y. Ma. Plane-based content preserving warps for video stabilization. In *2013 IEEE Conference on Computer Vision and Pattern Recognition, Portland, OR, USA, June 23-28, 2013*, pages 2299–2306. IEEE, 2013. doi: http://dx.doi.org/10.1109/CVPR.2013.298.

# Appendices

# Appendix A

# Computing geodesic distance transforms

Criminisi et al. (2010) use a raster scan method described by Toivanen (1996) to approximate the geodesic distance shown in equation 2.2. An overview of how the 2-dimensional geodesic distance transforms are calculated is shown, followed by an extension of the method for processing 3-dimensional video data. Criminisi et al. (2010) do not specify how they extend their raster scan method to 3-dimensions. The method implemented in chapter 2, for processing production footage, is detailed below.

## A.1 Calculating Geodesic Distances in 2-Dimensions

Toivanen (1996) employs a raster scan method which uses a 3x3 kernel (table A.1) to calculate the geodesic distance from each pixel in the image to the defined object area. The scans may be run more than once to improve the results of the distance transform.

| | | |
|---|---|---|
| a | b | c |
| d | e | f |
| g | h | k |

Table A.1: The 3x3 kernel

| | | |
|---|---|---|
| a | b | c |
| d | e | |
| | | |

Table A.2: The split kernel used in the first scan

**First Scan** The first scan runs from left-to-right, top-to-bottom using equation A.1 and the kernel shown in table A.2. $G(x)$ represents the original grey value of the image. $F(x)$ is the binary image showing the object area and is initialised to either 0 for the object area or a maximal value for the rest of the image. $F^*(x)$ denotes an already calculated point. In this scan $F^*(e)$ is the newly calculated point, while $F(e)$ contains the initial value from the binary image.

$$F^*(e) = min[F(e), min(da + F^*(a), db + F^*(b), \\ dc + F^*(c), dd + F^*(d))] \tag{A.1}$$

where

$$\begin{aligned} da &= \alpha\sqrt{(G(e) - G(a))^2 + \beta} \\ db &= \alpha\sqrt{(G(e) - G(b))^2 + \delta} \\ dc &= \alpha\sqrt{(G(e) - G(c))^2 + \beta} \\ db &= \alpha\sqrt{(G(e) - G(d))^2 + \delta} \end{aligned} \tag{A.2}$$

where $\delta = 1$ and $\beta = 2$.

| | | |
|---|---|---|
| | e | f |
| g | h | k |

Table A.3: The split kernel used in the second scan

**Second Scan** The second scan runs through the image data right-to-left, bottom-to-top using equation A.3 and the kernel shown in table A.3. In this second scan $F(e)$ contains the value calculated after applying equation A.1.

$$F^*(e) = min[F(e), min(df + F^*(f), dg + F^*(g), dh + F^*(h), dk + F^*(k))] \tag{A.3}$$

where

$$\begin{aligned} df &= \alpha\sqrt{(G(e) - G(f))^2 + \delta} \\ dg &= \alpha\sqrt{(G(e) - G(g))^2 + \beta} \\ dh &= \alpha\sqrt{(G(e) - G(h))^2 + \delta} \\ dk &= \alpha\sqrt{(G(e) - G(k))^2 + \beta} \end{aligned} \tag{A.4}$$

where $\delta = 1$ and $\beta = 2$.

Alternative weightings for $\delta$ and $\beta$ were suggested by Borgefors (1986) with $\delta = 0.95509$ and $\beta = 1.36930$.

## A.2 Calculating Geodesic Distances for a 3-Dimensional Video Cube

The transforms above work for 2-dimensional data such as a single image. However, for the research problem presented in chapter 2, an algorithm for multiple frames of video data was required. Therefore, the data had to be processed in 3-dimensions, with time (i.e. frame number) being the third dimension. The image frame data is batched together to form a video cube with axes of $x$ and $y$ for the pixel data and $t$ representing the frame numbers over time.

One extension of the 2-dimensional algorithm to work with the 3-dimensional data is shown below. It uses a 3x3x3 kernel as shown in table A.4. The table shows the current frame, the previous frame and the next frame, where $x_0$ represents the current frame, $x_{-1}$ represents the previous frame and $x_1$ represents the next frame. The method runs a 2 scan process in the same way as the single image scan.

| Previous frame | | | Current frame | | | Next frame | | |
|---|---|---|---|---|---|---|---|---|
| $a_{-1}$ | $b_{-1}$ | $c_{-1}$ | $a_0$ | $b_0$ | $c_0$ | $a_1$ | $b_1$ | $c_1$ |
| $d_{-1}$ | $e_{-1}$ | $f_{-1}$ | $d_0$ | $e_0$ | $f_0$ | $d_1$ | $e_1$ | $f_1$ |
| $g_{-1}$ | $h_{-1}$ | $k_{-1}$ | $g_0$ | $h_0$ | $k_0$ | $g_1$ | $h_1$ | $k_1$ |

Table A.4: The 3x3x3 kernel.

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| $a_{-1}$ | $b_{-1}$ | $c_{-1}$ | $a_0$ | $b_0$ | $c_0$ | | | |
| $d_{-1}$ | $e_{-1}$ | $f_{-1}$ | $d_0$ | $e_0$ | | | | |
| $g_{-1}$ | $h_{-1}$ | $k_{-1}$ | | | | | | |

Table A.5: The split 3x3x3 kernel used in the first scan.

**First Scan** The split kernel used in the first scan is shown in Table A.5. The first scan runs from left-to-right, top-to-bottom and front-to-back and processes the voxel data as shown in equation A.5.

$$\begin{aligned} F^*(e_0) = min[F(e_0), min( \quad & da_0 + F^*(a_0), db_0 + F^*(b_0), dc_0 + F^*(c_0), dd_0 + F^*(d_0), \\ & da_{-1} + F^*(a_{-1}), db_{-1} + F^*(b_{-1}), dc_{-1} + F^*(c_{-1}), \\ & dd_{-1} + F^*(d_{-1}), de_{-1} + F^*(e_{-1}), df_{-1} + F^*(f_{-1}), \\ & dg_{-1} + F^*(g_{-1}), dh_{-1} + F^*(h_{-1}), dk_{-1} + F^*(k_{-1}))] \end{aligned} \quad \text{(A.5)}$$

where

$$
\begin{aligned}
e.g. \qquad & da_0 = \alpha\sqrt{(G(e_0) - G(a_0))^2 + \beta} \\
& db_0 = \alpha\sqrt{(G(e_0) - G(b_0))^2 + \delta} \\
& da_{-1} = \alpha\sqrt{(G(e_0) - G(a_{-1}))^2 + \gamma}
\end{aligned}
\tag{A.6}
$$

where $\delta = 1$, $\beta = 2$ and $\gamma = 3$.

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | | | | $a_1$ | $b_1$ | $c_1$ |
| | | | | $e_0$ | $f_0$ | $d_1$ | $e_1$ | $f_1$ |
| | | | $g_0$ | $h_0$ | $k_0$ | $g_1$ | $h_1$ | $k_1$ |

Table A.6: The split 3x3x3 kernel used in the second scan.

**Second Scan** The split kernel used in the second scan of the video cube is shown in Table A.6. The second scan runs from right-to-left, bottom-to-top, and back-to-front using the equation A.7.

$$
\begin{aligned}
F^*(e_0) = min[F(e_0), min( \quad & df_0 + F^*(f_0), dg_0 + F^*(g_0), dh_0 + F^*(h_0), dk_0 + F^*(k_0) \\
& da_1 + F^*(a_1), db_1 + F^*(b_1), dc_1 + F^*(c_1), \\
& dd_1 + F^*(d_1), de_1 + F^*(e_1), df_1 + F^*(f_1), \\
& dg_1 + F^*(g_1), dh_1 + F^*(h_1), dk_1 + F^*(k_1))]
\end{aligned}
\tag{A.7}
$$

where

$$
\begin{aligned}
e.g. \qquad & df_0 = \alpha\sqrt{(G(e_0) - G(f_0))^2 + \delta} \\
& dg_0 = \alpha\sqrt{(G(e_0) - G(g_0))^2 + \beta} \\
& da_1 = \alpha\sqrt{(G(e_0) - G(a_1))^2 + \gamma}
\end{aligned}
\tag{A.8}
$$

where $\delta = 1$, $\beta = 2$ and $\gamma = 3$.

# Appendix B

# Improved image segmentation with motion: Nuke plug-in source code

The code in listing B.1 is the source code for the Nuke plug-in class which runs the video segmentation algorithm presented in chapter 2. It shows the main Nuke plug-in class, from which further classes are called to provide the functionality required by the algorithm. For example, the `GeodesicSegment` class contains the code to complete the geodesic operations.

```
1  // GeodesicVideoSegment.cpp
   // Author: Lindsey Howell
3
   #include "DDImage/Iop.h"
5  #include "DDImage/Row.h"
   #include "DDImage/Pixel.h"
7  #include "DDImage/Interest.h"
   #include "DDImage/Knobs.h"
9  #include <iostream>
   #include "DDImage/Memory.h"
11 #include <stdio.h>
   #include "DDImage/Thread.h"
13 #include <pthread.h>
   #include <sys/time.h>
15 #include "DDImage/ImageCache.h"   // *** Important - in the documentation called
       Image_Cache ***
                  // when instantiating the class use Image_Cache()
17 #include "DDImage/DDImage_API.h"

19 #include "highgui.h"
   #include "cv.h"
21


23 #include "GeodesicSegment.h"
   #include "FrequencyAnalysis.h"
25
```

```
using namespace DD::Image;

static const char* const CLASS = "GeodesicVideoSegment";
static const char* const HELP = "Segments the image based on the user foreground
    and background input - extended to work on video in a temporally stable way";


/*
 * GeodesicVideoSegment Plugin: Segments video frames based on the user foreground
     and background input.
 */
class GeodesicVideoSegment : public Iop
{

private:
  //overwritten methods
  void _validate(bool);
  virtual void _request(int, int, int, int, ChannelMask, int);
  virtual void engine(int y, int x, int r, ChannelMask, Row & t);
  virtual void _close();

  //threading methods...
  static void ggdtsForSigned(unsigned index, unsigned nThreads, void* d);
  static void /*Array3D<bool>**/ ggdtsForSymmetricSigned(unsigned index, unsigned
    nThreads, void* d);

  Array3D<bool>* geodesicSymmetricFilter(Array3D<float>* ssggdt);
  void runGeodesicVideoSegmentation(int fx, int fy, int fr, int ft, ChannelMask
    channels);
  void populateImageFrame(int framePosition, int frameObjectInput, int
    frameBackgroundInput, int frameMotionInput, int fx, int fy, int fr, int ft,
    ChannelMask channels);
  void deleteObjects();
  void checkCache(int fx, int fy, int fr, int ft, ChannelMask channels);

  //variables
  Video* _video;
  Channel _backgroundChannel;
  Channel _motionChannel;
  Array3D<float>*  _ssggdt;
  Array3D<bool>* _gsf;
  Array3D<float>* _finalOutputArray;
  Image* _outputImage;

  //optical flow additional variables
  float _opticalFlowStrength;

  bool _firstTime;
  Lock _lock;
  bool _firstFrame;
  time_t _startTime, _endTime;
```

```
71    timeval start, end;
      int _currentFrameVideoIndex;
73    bool cacheReadSuccess;

75    //user settable variables
      float _imageGradientStrength;
77    float _probabilityMapStrength;
      float _sigmoidTransformationFactor;
79    bool _initialiseWithProbMap;
      int _noRasterScans;
81    bool _runSuperScan;
      int _startFrameNumber, _endFrameNumber;
83    int _divergenceMethod;
      int _kernelSizeInput;
85    int _probMapType;
      float _colourMotionMix;
87    bool _useMinProbValue;
      bool _smoothProbHistogram;
89    bool _blurProbMap;
      int _blurKernelSize;
91    bool _fallOffProbMap;
      bool _distanceSquared;
93    bool _useProbMapPenalty;
      bool _usePolarCoords;
95    bool _autoMoveUserInputFG;
      bool _autoMoveUserInputBG;
97
      int _stageOfAlgorithm;
99    enum _stageEnum {ForegroundMask, ProbabilityMap, GeneralisedGDT, InverseGGDT,
        SignedGGDT, ErosionMask, DilationMask, ErosionGGDT, DilationGGDT,
        SymmetricSignedGGDT, GSFMask};


101
    public:
103   GeodesicVideoSegment(Node* node);
      virtual void knobs(Knob_Callback);
105   virtual int  knob_changed(Knob*);
      const char* Class() const { return CLASS; }
107   const char* node_help() const { return HELP; }
      static const Iop::Description d;
109   virtual void in_channels(int input, ChannelSet& channels) const;

111   //methods to handle the node inputs
      int minimum_inputs() const { return 1; }  //overwrite these members of Op class
113   int maximum_inputs() const { return 3; }
      //ensure all but one input is optional
115   int optional_input() const;
      //input labels
117   const char* input_label(int input, char* buffer) const
      {
119     switch (input)
```

```
    {
      case 0:
        return "obj";
      case 1:
        return "bkg";
      case 2:
        return "motion";
      default:
        return 0;
    }
  }

  //methods added to work with more than one frame at a time
  int split_input(int n) const;
  const OutputContext& inputContext(int, int, OutputContext&) const;

  //these are all put here so that they can be accessed by the threading methods
  GeodesicSegment* _geodesicSegment;
  Array3D<float>* _ggdt;
  Array3D<float>* _inverseGGDT;
  Array3D<float>* _signedGGDT;
  Array3D<float>* _erosionGGDT;
  Array3D<float>* _dilationGGDT;
  Array3D<bool>* _erosionMask;
  Array3D<bool>* _dilationMask;
  int _imageWidth, _imageHeight;
  int _erosionPixels, _dilationPixels;
  bool _swapErosionTrueFalse;

  int _numberOfFrames;

};


/*
 * Constructor - initialises the variables.
 */
GeodesicVideoSegment::GeodesicVideoSegment(Node* node) : Iop(node)
{
//  std::cout << "in constructor... \n";
  inputs();

    _backgroundChannel = Chan_Alpha;  //get only alpha values from the obj
    refinements channel

  //_firstTime = false;
  _firstFrame = true;

  //NB. set pointers to null
  _geodesicSegment = NULL;
  _video = NULL;
```

```
    _ssggdt = NULL;
    _gsf = NULL;
    _ggdt = NULL;
    _inverseGGDT = NULL;
    _signedGGDT = NULL;
    _erosionGGDT = NULL;
    _dilationGGDT = NULL;
    _finalOutputArray = NULL;
    _erosionMask = NULL;
    _dilationMask = NULL;
    _outputImage = NULL;

    _startTime = _endTime = 0;

    _startFrameNumber = 0;
    _endFrameNumber = 0;

    //default values for user-settable variables
    _usePolarCoords = false;
    _stageOfAlgorithm = 0;
    _smoothProbHistogram = false;
    _blurProbMap = false;
    _blurKernelSize = 1;
    _fallOffProbMap = false;
    _distanceSquared = false;
    _imageGradientStrength = 1;
    _useProbMapPenalty = false;
    _probabilityMapStrength = 10;
    _probMapType = 0;
    _useMinProbValue = false;
    _colourMotionMix = 0;
    _initialiseWithProbMap = true;
    _sigmoidTransformationFactor = 5;
    _noRasterScans = 2;
    _runSuperScan = false;
    _erosionPixels = 1;
    _swapErosionTrueFalse = true;
    _dilationPixels = 1;
    _startFrameNumber = 1;
    _endFrameNumber = 2;
    _opticalFlowStrength = 0;
    _divergenceMethod = 0;
    _kernelSizeInput = 1;
    _autoMoveUserInputFG = false;
    _autoMoveUserInputBG = false;
}
```

```
221
/*
223  * return the number of NON-optional inputs
 */
225 int GeodesicVideoSegment::optional_input() const
{
227   return 1;  // for this operator there is 1 input that is not optional.
}
229
// Methods to access more than one frame input - must be overwritten.
231 int GeodesicVideoSegment::split_input(int inputNo) const
{
233   //Split the input into the number of frames that need to be read
   //add one as you want the frame range to be inclusive.
235   return _endFrameNumber - _startFrameNumber + 1;
}
237

239 int _currentFrame;
const OutputContext& GeodesicVideoSegment::inputContext(int n, int offset,
    OutputContext& context) const
241 {
   context = outputContext();
243   _currentFrame = (int)context.frame();

245   //if it's not in the frame range then don't need to run the segmentation
     procedure - just process the one frame individually
   if ((_currentFrame < _startFrameNumber) || (_currentFrame > _endFrameNumber))
247   {
     return context;
249   }
   else  //if it is in the frame range then want to get all frames, starting with
     the current frame
251   {
     if (offset == 0) // the current frame
253     {
       return context;
255     }
     else
257     {
       if (_startFrameNumber + offset - 1 < _currentFrame)
259       {
         context.setFrame(_startFrameNumber + offset - 1);
261       }
       else if (_startFrameNumber + offset > _currentFrame)
263       {
         context.setFrame(_startFrameNumber + offset);
265       }
       else
267       {
         std::cout << "*** Error in inputContext - invalid frame range. \n";
```

```
269       }
          return context;
271     }
      }
273   return context;
}
275

277 /*
 * Perform error checks before the engine runs
279  */
void GeodesicVideoSegment::_validate(bool)
281 {
    if (_imageGradientStrength < 0)
283   {
      Op::error("The Image Gradient Strength must be greater than 0.");
285   }
    if (_probabilityMapStrength < 0)
287   {
      Op::error("The Probability Map Strength must be zero or above.");
289   }
    if (_noRasterScans < 1)
291   {
      Op::error("You must set the raster scans to run at least once.");
293   }
    if ((_sigmoidTransformationFactor < 1) || (_sigmoidTransformationFactor > 10))
295   {
      _sigmoidTransformationFactor = 5;
297   Op::error("You must set the sigmoid transformation factor to be between 1 and
      10.");
    }
299 if (_erosionPixels <= 0)
    {
301   Op::error("Erosion pixels must be above zero.");
    }
303 if (_dilationPixels <= 0)
    {
305   Op::error("Dilation pixels must be above zero.");
    }
307
    if (_startFrameNumber < 0)
309 {
      Op::error("Start Frame must be greater than zero.");
311 }
    if (_endFrameNumber < 0)
313 {
      Op::error("End Frame must be greater than zero.");
315 }
    if (_startFrameNumber > _endFrameNumber)
317 {
      Op::error("End Frame must be greater than the Start Frame.");
```

```
319   }
      if (_colourMotionMix < 0 || _colourMotionMix > 1)
321   {
        Op::error("The colour to motion mix must be between 0 and 1.");
323   }
      if (_blurKernelSize < 1 || _blurKernelSize > 5)
325   {
        Op::error("The blur kernel size must be between 1 and 5");
327   }
      copy_info();
329
      _numberOfFrames = _endFrameNumber - _startFrameNumber + 1;
331   for (int i = 0; i < _numberOfFrames; i++)
      {
333     if (input((2*_numberOfFrames)+i) != 0)
        {
335       merge_info((2*_numberOfFrames)+i); //merge the motion information channels
        so they can be accessed.
        }
337   }

339   //set which channels to output from the node
      ChannelSet outChannels;
341   outChannels += Mask_All;
      outChannels += Chan_U;
343   outChannels += Chan_V;
      outChannels += Chan_Backward_U;
345   outChannels += Chan_Backward_V;
      set_out_channels(outChannels);
347
      _firstTime = true;
349 }

351 void GeodesicVideoSegment::in_channels(int input, ChannelSet& channels) const
    {
353   //add the motion vector channels to the channels that the plugin gets
      channels += Mask_MoVec;
355 }


357
    /*
359  * Get the information needed from the image row
     */
361 ChannelSet in;
    void GeodesicVideoSegment::_request(int x, int y, int r, int t, ChannelMask
        channels, int count)
363 {
      for (int n = 0; n < inputs(); n++)
365   {
        Iop * thisInput = dynamic_cast<Iop*>(Op::input(n));
367     in = channels;
```

```
    in_channels(n, in);
    if (thisInput && in)
    {
      thisInput->request(in, 0);
    }
  }


  _firstTime = true;

  for (int i = 0; i < _numberOfFrames; i++)
  {
    //need to get foreground and background inputs for each frame
    input(i)->request(x, y, r, t, channels, count); //gets the pixel input ready
    for use in the engine
    if (input(_numberOfFrames+i) != 0)
    {
      input(_numberOfFrames+i)->request(x, y, r, t, _backgroundChannel, count); //
    get the input describing refinements to the object area
    }
    // *** Add input for motion channel here...
    if (input((2*_numberOfFrames)+i) != 0)
    {
      input((2*_numberOfFrames)+i)->request(x, y, r, t, channels, count);
    }
  }
}



/*
 * Performs the main image processing
 */
void GeodesicVideoSegment::engine(int y, int x, int r, ChannelMask channelMask,
    Row& row)
{
  ChannelSet channels = channelMask;

  //initial analysis to get source image - only happens the first time the engine
    is called.
  Guard guard(_lock);
  if ( _firstTime )
  {
    channels += in;

    //start the clock running, for testing
    _startTime = clock();
      gettimeofday(&start, 0);

    //format the image variables for getting the interest area
    Format format = input(0)->format();
```

```
415     const int fx = format.x();
        const int fy = format.y();
417     const int fr = format.r();
        const int ft = format.t();
419     _imageWidth = fr - fx;
        _imageHeight = ft - fy;
421
        //calculate where the current frame is in the video sequence that's stored
423     _currentFrameVideoIndex = _currentFrame - _startFrameNumber;

425     if ((_currentFrame >= _startFrameNumber) && (_currentFrame <= _endFrameNumber)
        )
        {
427       //check whether the output for the current settings is stored in the cache
          checkCache(fx, fy, fr, ft, channels);
429
          float maximumValue = maximum(_finalOutputArray->getData(), _finalOutputArray
        ->getArrayDimension(Width)*_finalOutputArray->getArrayDimension(Height)*
        _finalOutputArray->getArrayDimension(Depth));
431       float minimumValue = minimum(_finalOutputArray->getData(), _finalOutputArray
        ->getArrayDimension(Width)*_finalOutputArray->getArrayDimension(Height)*
        _finalOutputArray->getArrayDimension(Depth));
          if (maximumValue == 0) maximumValue = 1;
433       if (minimumValue == 0) minimumValue = 1;
          //get output into correct range - normalise -1->0->1, put negative in red,
        positive in green.
435       _outputImage = new Image(_imageWidth, _imageHeight);
          for (int row = 0; row < _imageHeight; row++)
437       {
            for (int col = 0; col < _imageWidth; col++)
439         {
              if (_finalOutputArray->getArrayValue(row, col, _currentFrameVideoIndex)
        < 0)
441           {
                (*_outputImage)(row, col).r = (_finalOutputArray->getArrayValue(row,
        col, _currentFrameVideoIndex)/minimumValue);
443           }
              else
445           {
                (*_outputImage)(row, col).g = _finalOutputArray->getArrayValue(row,
        col, _currentFrameVideoIndex)/maximumValue;
447           }

449       }
          }
451


453       //end the clock here as this is after main processing is done...
          _endTime = clock();
455
          gettimeofday(&end, 0);
```

```
457       long seconds  = end.tv_sec  - start.tv_sec;
          long useconds = end.tv_usec - start.tv_usec;
459       double duration = seconds + useconds/1000000.0;
          if (!cacheReadSuccess)
461       {
            std::cout << "Time taken: " << duration << std::endl;
463       }
        }
465
        _firstTime = false;
467

469   }  //*** end lock ***

471
    // OUTPUT PIXELS
473
    // if the current frame isn't within the range that should be segmented then
        output as is...
475 if ((_currentFrame < _startFrameNumber) || (_currentFrame > _endFrameNumber))
    {
477   //Output the pixels without any changes
      foreach(z, channels)
479   {
        float* outptr = row.writable(z) + x;
        for(int i = x; i < r; i++)
481     {
          const char* channelName = getName(z);
483       if ((strcmp(channelName, "forward.u") == 0) )
          {
485         Pixel original_pixel(z);
            if (input(2*_numberOfFrames) != 0)
487         {
              input(2*_numberOfFrames)->at(i, y, original_pixel);
489           *outptr++ = (float)(original_pixel[z]);
            }
491         else
            {
493           *outptr++ = 0;
            }
495       }
          else
497       {
            Pixel original_pixel(z);
499         input(0)->at(i, y, original_pixel);
            *outptr++ = original_pixel[z];
501       }
        }
503   }
    }
505 }
    else
```

```
507   {
        //otherwise, if it's in the frame-range, then output the result
509     foreach(z, channels)
        {
511       float* outptr = row.writable(z) + x;
          for(int i = x; i < r; i++)
513       {
            switch (colourIndex(z))
515         {
              case 0:
517             *outptr++ = (*_outputImage)(y, i).r;
                break;
519           case 1:
                *outptr++ = (*_outputImage)(y, i).g;
521             break;
              case 2:
523             *outptr++ = (*_outputImage)(y, i).b;
                break;
525           case 3:
                if ((*_outputImage).foregroundMask(y, i) == true)
527             {
                  *outptr++ = 1;
529             }
                else
531             {
                  *outptr++ = 0;
533             }
                break;
535           default:
                Pixel original_pixel(z);
537             input(0)->at(i, y, original_pixel);
                *outptr++ = original_pixel[z];
539             break;
            }
541       }
        }
543   }
}
545


547

/*
549  *  Checks the cache and only run the segmentation code if the results aren't
         cached
     *  for the current parameters (user variables, inputs, resolution).
551  *
     */
553 void GeodesicVideoSegment::checkCache(int fx, int fy, int fr, int ft, ChannelMask
         channels)
{
555   Image_Cache *iCache = &Image_Cache::mainCache();
```

```
557   //create the hash object...
      DD::Image::Hash hash;
559   hash.reset();
      //append correct information for this hash key - user parameters, resolution
        size and input nodes
561   hash.append(_imageGradientStrength);
      hash.append(_probabilityMapStrength);
563   hash.append(_sigmoidTransformationFactor);
      hash.append(_initialiseWithProbMap);
565   hash.append(_noRasterScans);
      hash.append(_runSuperScan);
567   hash.append(_erosionPixels);
      hash.append(_dilationPixels);
569   hash.append(_startFrameNumber);
      hash.append(_endFrameNumber);
571   hash.append(_opticalFlowStrength);
      hash.append(_divergenceMethod);
573   hash.append(_kernelSizeInput);
      hash.append(_probMapType);
575   hash.append(_colourMotionMix);
      hash.append(_useMinProbValue);
577   hash.append(_swapErosionTrueFalse);
      hash.append(_smoothProbHistogram);
579   hash.append(_blurProbMap);
      hash.append(_blurKernelSize);
581   hash.append(_stageOfAlgorithm);
      hash.append(_fallOffProbMap);
583   hash.append(_distanceSquared);
      hash.append(_useProbMapPenalty);
585   hash.append(_usePolarCoords);
      hash.append(_autoMoveUserInputFG);
587   hash.append(_autoMoveUserInputBG);
      hash.append(outputContext().scale_x()); //for downrez issue - needed it to
        recalculate on downrez
589   hash.append(outputContext().scale_y());

591   //get the hashes for the input nodes
      //need to get them in the same order each time so that the hash key doesn't
        change with frame changes
593   for (int i = _startFrameNumber; i <= _endFrameNumber; i++)
      {
595     //once the image object (frame) has been created then add it to the video
        object in the correct place
        int inputNo;
597     if (i < _currentFrame) // the current frame
        {
599       inputNo = i - _startFrameNumber + 1;
        }
601     else if (i == _currentFrame)
        {
```

```
603       inputNo = 0;
      }
605     else if (i > _currentFrame)
      {
607       inputNo = i - _startFrameNumber;
      }
609
      //add the hash keys of the input nodes to the hash
611   //main input for current frame
      hash.append(input(inputNo)->hash().getHash());
613   //background mask input for current frame
      if (input(inputNo + _numberOfFrames)  != 0)
615   {
        hash.append(input(inputNo + _numberOfFrames)->hash().getHash());
617   }
      //also need to add motion mask here...
619   if (input(inputNo + (2*_numberOfFrames)) != 0)
      {
621     hash.append(input(inputNo + (2*_numberOfFrames))->hash().getHash());
      }
623 }

625 //... then check to see if this hashkey is in the cache
    cacheReadSuccess = true;
627 if (iCache->is_active() && iCache->has_file(hash))
    {
629   ImageCacheReadI* cacheRead = iCache->open(hash); //for v6.3!
      // *** changed for version 6.2 ***
631   //ImageCacheRead* cacheRead = iCache->open(hash);

633   float *videoObjFromCache = new float[_numberOfFrames * _imageWidth *
      _imageHeight];
      size_t desiredNoReadBytes = (_numberOfFrames * _imageWidth * _imageHeight) *
      sizeof(float);
635
      //read the object from the cache
637   // *** changed for version 6.2 ***
      size_t readBytes = cacheRead->read(videoObjFromCache, desiredNoReadBytes); //
      for v6.3!
639   //size_t readBytes = iCache->read(videoObjFromCache, desiredNoReadBytes,
      cacheRead);

641   if (!iCache->is_read() || readBytes != desiredNoReadBytes)
      {
643     cacheReadSuccess = false;
      }
645   else
      {
647     //create new video object to hold each of the frames
        _finalOutputArray = new Array3D<float>(_imageWidth, _imageHeight,
      _numberOfFrames);
```

```
649
      //read in all the float values in the array read from the cache into the
    _video object
651   int arrayIndex = 0;
      for (int frameNumber = 0; frameNumber < _numberOfFrames; frameNumber++)
653   {
        for (int row = 0; row < _imageHeight; row++)
655     {
          for (int col = 0; col < _imageWidth; col++)
657       {
            //add each of the colour values
659         _finalOutputArray->setArrayValue(row, col, frameNumber,
    videoObjFromCache[arrayIndex]);
            arrayIndex ++;
661       }
        }
663   }
    }
665
    cacheRead->close(); //  for 6.3!
667 // *** changed for version 6.2 ***
    //iCache->close(cacheRead);
669
    //free memory
671 delete[] videoObjFromCache;
  }
673 else
  {
675 cacheReadSuccess = false;
  }
677
  // if the object could not be read from cache (either it wasn't in there or the
    operation failed)...
679 if (!cacheReadSuccess)
  {
681 std::cout << "* cache has not been read so video cube is calculated *\n";

683 //calculate the value of the video cube object
    runGeodesicVideoSegmentation(fx, fy, fr, ft, channels);
685
    //set up writer to write to the cache
687 // *** changed for version 6.2 ***
    DD::Image::ImageCacheWriteI* cacheWrite = iCache->create(hash); //for v6.3!
689 //DD::Image::ImageCacheWrite* cacheWrite = iCache->create(hash);

691 //and then create the object with the information that wants to be written to
    the cache
    float *videoObjToCache = new float[_numberOfFrames * _imageWidth *
    _imageHeight * 3];
693 //populate the float array with all the values in the video obj
    int arrayIndex = 0;
```

```
695     for (int frameNumber = 0; frameNumber < _numberOfFrames; frameNumber++)
        {
697       for (int row = 0; row < _imageHeight; row++)
          {
699         for (int col = 0; col < _imageWidth; col++)
            {
701           videoObjToCache[arrayIndex] = /*_ssggdt*/ _finalOutputArray->
        getArrayValue(row, col, frameNumber);
              arrayIndex++;
703         }
          }
705     }

707     // And write the array to cache
        size_t desiredNoWriteBytes = (_numberOfFrames * _imageWidth * _imageHeight) *
        sizeof(float);
709     // *** changed for version 6.2 ***
        cacheWrite->write(videoObjToCache, desiredNoWriteBytes);  //for v6.3!
711     //iCache->write(videoObjToCache, desiredNoWriteBytes, cacheWrite);

713     //free memory
        delete[] videoObjToCache;
715
        if (!iCache->is_written())
717     {
          std::cout << "Error saving video cube to cache. Have written: " << (int)
        iCache->is_written() << "\n";
719     }

721     // *** changed for version 6.2 ***
        cacheWrite->close();  //for v6.3!
723     //iCache->close(cacheWrite);
    }
725 }

727 /*
     * This method filters the results of the symmetric signed generalised geodesic
        distance transform
729  * to give the final segmentation.
     * Have moved this method from the GeodesicSegment class so that it can still be
        called even if the
731  * object is fetched from the cache.
     */
733 Array3D<bool>* GeodesicVideoSegment::geodesicSymmetricFilter(Array3D<float>*
        ssggdt)
    {
735   //create the array to put the results in
      Array3D<bool>* gsf = new Array3D<bool>(_imageWidth, _imageHeight,
        _numberOfFrames);
737   for (int frameNumber = 0; frameNumber < _numberOfFrames; frameNumber++)
      {
```

```
739     for (int row = 0; row < _imageHeight; row++)
        {
741       for (int col = 0; col < _imageWidth; col++)
          {
743         if (ssggdt->getArrayValue(row, col, frameNumber) < 0)
            {
745           gsf->setArrayValue(row, col, frameNumber, true);
            }
747         else
            {
749           gsf->setArrayValue(row, col, frameNumber, false);
            }
751       }
      }
753   }
    return gsf;
755 }

757

759 /* This function creates the video object and applies the geodesic distance
        transform
     * and filtering code to it. It results in a 3D array containing the segmented
        video frames.
761  */
    void GeodesicVideoSegment::runGeodesicVideoSegmentation(int fx, int fy, int fr,
        int ft, ChannelMask channels)
763 {
    std::cout << "at start of runGeodesicVideoSeg \n";
765 //create new video object to hold each of the frames
    _video = new Video(_numberOfFrames, _imageWidth, _imageHeight/*, _noFreqBands*/)
        ;
767
    int frameObjectInput = 0;
769 int frameBackgroundInput = _numberOfFrames;
    int frameMotionInput = 2 * _numberOfFrames;
771 if ((_currentFrame < _startFrameNumber) || (_currentFrame > _endFrameNumber))
    {
773   std::cout << "outside the frame range \n";
    }
775 else
    {
777   //add a loop to run through each of the frames and assign them to the video
      object accordingly
      for (int i = 0; i < _numberOfFrames; i++)
779   {
        frameObjectInput = i;
781     // as it is accessing more than one frame, the background input channel now
      comes after each frame's main channel
        frameBackgroundInput = _numberOfFrames + i;
783     frameMotionInput = (2*_numberOfFrames) + i;
```

```
785       //once the image object (frame) has been created then add it to the video
    object in the correct place
      int framePosition;
787       if (i == 0) // the current frame
      {
789         framePosition = _currentFrame;
      }
791       else
      {
793         if (_startFrameNumber + i - 1 < _currentFrame)
          {
795           framePosition = _startFrameNumber + i - 1;
          }
797         else if (_startFrameNumber + i > _currentFrame)
          {
799           framePosition = _startFrameNumber + i;
          }
801       }

803       //update the values in the video object for that frame...
          //need to subtract the _startFrameNumber so that it is entered into the
    video frames at 0
805       populateImageFrame(framePosition - _startFrameNumber, frameObjectInput,
    frameBackgroundInput, frameMotionInput, fx, fy, fr, ft, channels);
    }
807
    //Now that the images are populated...
809 //calculate minimum and maximum values for the x and y motions
    _video->calculateMinMaxMotions();
811
    //Now that images are populated, calculated texture information and store for
    making histograms
813 _video->calculateTextureValues();

815 }
  std::cout << "before create GeodesicSegment \n";
817
  //create a geodesicSegment object to perform operations on the image
819 _geodesicSegment = new GeodesicSegment(_video, input(frameBackgroundInput) != 0,
      _imageGradientStrength, _probabilityMapStrength,
    _sigmoidTransformationFactor,
821   _noRasterScans, _erosionPixels, _dilationPixels, _initialiseWithProbMap,
    _runSuperScan,
      _opticalFlowStrength, _divergenceMethod, _kernelSizeInput, _probMapType, /*
    _showOnlyProbMap,*/
823   _colourMotionMix, _useMinProbValue, _smoothProbHistogram, _blurProbMap,
    _blurKernelSize,
      _fallOffProbMap, _distanceSquared, _useProbMapPenalty);
825
```

```
827   switch(_stageOfAlgorithm)
      {
829     case ForegroundMask:
        {
831       //This one only works if you clear the cache, as it relies on the
      foregroundMask being populated.
        _finalOutputArray = new Array3D<float>(_imageWidth, _imageHeight,
      _numberOfFrames);
833       for (int frameNumber = 0; frameNumber < _numberOfFrames; frameNumber++)
          {
835         for (int row = 0; row < _imageHeight; row++)
            {
837           for (int col = 0; col < _imageWidth; col++)
              {
839             (_video->foregroundMask(row, col, frameNumber) == false) ?
      _finalOutputArray->setArrayValue(row, col, frameNumber, 0.0)
                  : _finalOutputArray->setArrayValue(row, col, frameNumber, 1.0);
841           }
            }
843       }
          break;
845     }
      case ProbabilityMap:
847     _finalOutputArray = _geodesicSegment->calculateProbabilityMap(true);
        break;
849   case GeneralisedGDT:
        ggdtsForSigned(0, 2, this);  //just call the method separately to generate
      only the _ggdt
851     _finalOutputArray = _ggdt;
        break;
853   case InverseGGDT:
        ggdtsForSigned(1, 2, this);
855     _finalOutputArray = _inverseGGDT;
        break;
857   case SignedGGDT:
        Thread::spawn(ggdtsForSigned, 2, this);
859     Thread::wait(this);
        _finalOutputArray = _geodesicSegment->signedGGDT(_ggdt, _inverseGGDT);
861     break;
      case ErosionMask:
863   {
        Thread::spawn(ggdtsForSigned, 2, this);
865     Thread::wait(this);
        //calculate the symmetric signed generalised geodesic distance transform
867     _signedGGDT = _geodesicSegment->signedGGDT(_ggdt, _inverseGGDT);
        ggdtsForSymmetricSigned(0, 2, this); //0 for erosion mask
869     _finalOutputArray = new Array3D<float>(_imageWidth, _imageHeight,
      _numberOfFrames);
        for (int frameNumber = 0; frameNumber < _numberOfFrames; frameNumber++)
871     {
          for (int row = 0; row < _imageHeight; row++)
```

```
873         {
              for (int col = 0; col < _imageWidth; col++)
875           {
                (_erosionMask->getArrayValue(row, col, frameNumber) == true) ?
    _finalOutputArray->setArrayValue(row, col, frameNumber, 0.0)
877                 : _finalOutputArray->setArrayValue(row, col, frameNumber, 1.0);
              }
879         }
          }
881       break;
        }
883     case DilationMask:
        {
885       Thread::spawn(ggdtsForSigned, 2, this);
          Thread::wait(this);
887       //calculate the symmetric signed generalised geodesic distance transform
          _signedGGDT = _geodesicSegment->signedGGDT(_ggdt, _inverseGGDT);
889       ggdtsForSymmetricSigned(1, 2, this); //1 for dilation mask
          _finalOutputArray = new Array3D<float>(_imageWidth, _imageHeight,
    _numberOfFrames);
891       for (int frameNumber = 0; frameNumber < _numberOfFrames; frameNumber++)
          {
893         for (int row = 0; row < _imageHeight; row++)
            {
895           for (int col = 0; col < _imageWidth; col++)
              {
897             (_dilationMask->getArrayValue(row, col, frameNumber) == true) ?
    _finalOutputArray->setArrayValue(row, col, frameNumber, 0.0)
                  : _finalOutputArray->setArrayValue(row, col, frameNumber, 1.0);
899           }
            }
901       }
          break;
903     }
        case ErosionGGDT:
905       Thread::spawn(ggdtsForSigned, 2, this);
          Thread::wait(this);
907       _signedGGDT = _geodesicSegment->signedGGDT(_ggdt, _inverseGGDT);
          Thread::spawn(ggdtsForSymmetricSigned, 2, this);
909       Thread::wait(this);
          _finalOutputArray = _erosionGGDT;
911       break;
        case DilationGGDT:
913       Thread::spawn(ggdtsForSigned, 2, this);
          Thread::wait(this);
915       _signedGGDT = _geodesicSegment->signedGGDT(_ggdt, _inverseGGDT);
          Thread::spawn(ggdtsForSymmetricSigned, 2, this);
917       Thread::wait(this);
          _finalOutputArray = _dilationGGDT;
919       break;
        case SymmetricSignedGGDT:
```

```
921       Thread::spawn(ggdtsForSigned, 2, this);
          Thread::wait(this);
923       _signedGGDT = _geodesicSegment->signedGGDT(_ggdt, _inverseGGDT);
          Thread::spawn(ggdtsForSymmetricSigned, 2, this);
925       Thread::wait(this);
          _finalOutputArray = _geodesicSegment->signedSymmetricGGDT(_erosionGGDT,
        _dilationGGDT);
927       break;
        case GSFMask:
929     {
          Thread::spawn(ggdtsForSigned, 2, this);
931       Thread::wait(this);
          _signedGGDT = _geodesicSegment->signedGGDT(_ggdt, _inverseGGDT);
933       Thread::spawn(ggdtsForSymmetricSigned, 2, this);
          Thread::wait(this);
935       _ssggdt = _geodesicSegment->signedSymmetricGGDT(_erosionGGDT, _dilationGGDT)
        ;
          _gsf = geodesicSymmetricFilter(_ssggdt);
937       _finalOutputArray = new Array3D<float>(_imageWidth, _imageHeight,
        _numberOfFrames);
          for (int frameNumber = 0; frameNumber < _numberOfFrames; frameNumber++)
939       {
            for (int row = 0; row < _imageHeight; row++)
941         {
              for (int col = 0; col < _imageWidth; col++)
943           {
                (_gsf->getArrayValue(row, col, frameNumber) == true) ?
        _finalOutputArray->setArrayValue(row, col, frameNumber, 0.0)
945                 : _finalOutputArray->setArrayValue(row, col, frameNumber, 1.0);
              }
947         }
          }
949       break;
      }
951   default:
        std::cout << "**** Shouldn't get to here - in the _stageOfAlgorithm switch
        statement. *****\n";
953     _finalOutputArray = new Array3D<float>(_imageWidth, _imageHeight,
      _numberOfFrames);
        _finalOutputArray->initialiseArray(0);
955     break;
    }
957 }


959
  /*
961  * This method populates the values of a single frame in the video cube object
   * given the frame position (in the video cube) and the inputs from which it reads
963  * data to populate the frame.
   */
965 void GeodesicVideoSegment::populateImageFrame(int framePosition, int
```

```
    frameObjectInput, int frameBackgroundInput, int frameMotionInput, int fx, int
    fy, int fr, int ft, ChannelMask channels)
{
  //Interest area needs to be of the selected source image area.
  if (input(frameObjectInput) != 0)
  {
    Interest interest(*input(frameObjectInput), fx, fy, fr, ft, channels, true);
    interest.unlock();
  }

  if (input(frameBackgroundInput) != 0)
  {
    Interest bkgInterest(*input(frameBackgroundInput), fx, fy, fr, ft,
    _backgroundChannel, true);
    bkgInterest.unlock();
  }
  if (input(frameMotionInput) != 0)
  {
    Interest motionInterest(*input(frameMotionInput), fx, fy, fr, ft, channels,
    true);
    motionInterest.unlock();
  }

  Image* _sourceImage = _video->_imageFrames[framePosition];
  float R, G, B, A, backgroundAlpha;

  //MotionVectorImage _divergenceImage = new MotionVectorImage(_imageWidth,
    _imageHeight, Forwards);
  float forwardMotionX, forwardMotionY, backwardMotionX, backwardMotionY;

  //loop through whole image to populate the sourceImage object
  for (int ry = fy; ry < ft; ry++)
  {
    //for every pixel in the row... populate the source image
    for(int i = fx; i < fr; i++)
    {
      //for each channel, read the pixel values
      foreach(z, channels)
      {
        Pixel original_pixel(z);
        //get the original pixel value from the image
        input(frameObjectInput)->at(i, ry, original_pixel);
        //assign the RGB values to the variables
        if  (strcmp(getName(z), "rgba.red") == 0)
        {
          R = (float)(original_pixel[z]);
        }
        else if (strcmp(getName(z), "rgba.green") == 0)
        {
          G = (float)(original_pixel[z]);
        }
```

```
        else if (strcmp(getName(z), "rgba.blue") == 0)
        {
          B = (float)(original_pixel[z]);
        }
        else if (strcmp(getName(z), "rgba.alpha") == 0)
        {
          A = (float)(original_pixel[z]); // the alpha is the foreground mask
    input
        }


        if (input(frameBackgroundInput) != 0)
        {
          //get the background mask information
          input(frameBackgroundInput)->at(i, ry, original_pixel);
          //if (colourIndex(z) == 3)
          if (strcmp(getName(z), "rgba.alpha") == 0)
          {
            backgroundAlpha = (float)(original_pixel[z]);
          }
        }

        if (input(frameMotionInput) != 0)
        {
          Pixel original_pixel(z);
          const char* channelName = getName(z);
          input(frameMotionInput)->at(i, ry, original_pixel);

          if (strcmp(getName(z), "forward.u") == 0)
          {
            forwardMotionX = (float)(original_pixel[z]);
          }
          else if (strcmp(getName(z), "forward.v")  == 0)
          {
            forwardMotionY = (float)(original_pixel[z]);
          }
          if (strcmp(getName(z), "backward.u") == 0)
          {
            backwardMotionX = (float)(original_pixel[z]);
          }
          else if (strcmp(getName(z), "backward.v")  == 0)
          {
            backwardMotionY = (float)(original_pixel[z]);
          }
        }
      }

       //populate Source Image here with the RGB values, foreground mask values and
      background mask values
       (*_sourceImage)(ry, i) = Color(R,G,B);
```

```
1061        //set the alpha values for the object area in the image.
            if ((A > 0.25)
1063          || (_sourceImage->foregroundMask(ry, i) == true))  //also check in case
        the motion has already been moved forward...
            {
1065          _sourceImage->foregroundMask(ry, i) = true;

1067          if ((_autoMoveUserInputFG) && (framePosition < _numberOfFrames-1))
              {
1069            // Carry the user input forward one frame on each call of this method.
                //get the next frame
1071            Image* _nextImage = _video->_imageFrames[framePosition+1];
                //and move the input mask using the motion vector information...
1073            int newX = 0, newY = 0;
                newX = (int)floor(i + forwardMotionX);
1075            newY = (int)floor(ry + forwardMotionY);
                _nextImage->foregroundMask(newY, newX) = true;
1077          }
            }
1079        else
            {
1081          _sourceImage->foregroundMask(ry, i) = false;
            }
1083
            if (input(frameBackgroundInput) != 0)
1085        {
              //set the alpha values for the background area in the image.
1087          if ((backgroundAlpha > 0.25)
                || (_sourceImage->backgroundMask(ry, i) == true))  //also check in case
        the motion has already been moved forward...
1089          {
                _sourceImage->backgroundMask(ry, i) = true;
1091
                if ((_autoMoveUserInputBG) && (framePosition < _numberOfFrames-1))
1093            {
                  // Start with carrying the user input forward one frame.
1095              //get the next frame
                  Image* _nextImage = _video->_imageFrames[framePosition+1];
1097              //and move the input mask using the motion vector information...
                  int newX = 0, newY = 0;
1099              newX = (int)floor(i + forwardMotionX);
                  newY = (int)floor(ry + forwardMotionY);
1101              _nextImage->backgroundMask(newY, newX) = true;
                }
1103          }
              else
1105          {
                _sourceImage->backgroundMask(ry, i) = false;
1107          }
            }
1109
```

```
      //set the motion values into the motion vector image object
      if ((forwardMotionX != 0) || (forwardMotionY != 0))
      {
        if (_usePolarCoords)
        {
          float lengthForward = sqrt(pow(forwardMotionX, 2) + pow(forwardMotionY,
    2));
          float rotationForward = atan(forwardMotionY/forwardMotionX);
          float lengthBackward = sqrt(pow(backwardMotionX, 2) + pow(
    backwardMotionY, 2));
          float rotationBackward = atan(backwardMotionY/backwardMotionX);
          _sourceImage->forwardMotion(ry, i) = MotionVector(lengthForward,
    rotationForward);
          _sourceImage->backwardMotion(ry, i) = MotionVector(lengthBackward,
    rotationBackward);
        }
        else
        {
          _sourceImage->forwardMotion(ry, i) = MotionVector(forwardMotionX,
    forwardMotionY);
          _sourceImage->backwardMotion(ry, i) = MotionVector(backwardMotionX,
    backwardMotionY);
        }
      }
    }
  }

  //calculate the greyscale of the image in order to use it in the distance
    transform
  _sourceImage->calculateGreyscale();
}


/*
 * Called at the end - once all rows have been processed.
 */
void GeodesicVideoSegment::_close()
{
    //delete any global objects created during the algorithm
  deleteObjects();
}

/*
 * Delete the pointer variables that have been used.
 */
void GeodesicVideoSegment::deleteObjects()
{
  if (_ssggdt)
  {
    delete _ssggdt;
    _ssggdt = NULL;
```

```
  }

  if (_gsf)
  {
    delete _gsf;
    _gsf = NULL;
  }

  if (_ggdt)
  {
    delete _ggdt;
    _ggdt = NULL;
  }
  if (_inverseGGDT)
  {
    delete _inverseGGDT;
    _inverseGGDT = NULL;
  }
  if (_signedGGDT)
  {
    delete _signedGGDT;
    _signedGGDT = NULL;
  }
  if (_erosionGGDT)
  {
    delete _erosionGGDT;
    _erosionGGDT = NULL;
  }
  if (_dilationGGDT)
  {
    delete _dilationGGDT;
    _dilationGGDT = NULL;
  }
  if (_geodesicSegment)
  {
    delete _geodesicSegment;
    _geodesicSegment = NULL;
  }
  if (_video)
  {
    delete _video;
    _video = NULL;
  }

  if (_finalOutputArray)
  {
    //the finalOutputArray is only created as new for some of the stages of the
    algorithm
    if (_stageOfAlgorithm == (GSFMask || ErosionMask || DilationMask))
    {
      delete _finalOutputArray;  // **** still causing seg fault since adding in
```

```
    the extra blur method in GeodesicSegment
    }
1205  _finalOutputArray = NULL;
  }
1207
  if (_erosionMask)
1209  {
    delete _erosionMask;
1211  _erosionMask = NULL;
  }
1213
  if (_dilationMask)
1215  {
    delete _dilationMask;
1217  _dilationMask = NULL;
  }
1219
  if (_outputImage)
1221  {
    delete _outputImage;
1223  _outputImage = NULL;
  }
1225
  //remove any memory left unused by Nuke
1227  Memory::reduce_current_usage();

1229  _firstTime = true;
}
1231
/*
1233 * Create the user controls
 */
1235 void GeodesicVideoSegment::knobs(Knob_Callback f)
{
1237  Bool_knob(f, &_usePolarCoords, "Use polar co-ordinates");
  SetFlags(f, Knob::STARTLINE);
1239
  const char* stageLabels[] = {"Input FG Mask", "Probability map", "Generalised
    Geodesic Distance Transform", "Inverse GGDT", "Signed GGDT",
1241    "Erosion Mask", "Dilation Mask", "Erosion GGDT", "Dilation GGDT", "Signed
    Symmetric GGDT", "Geodesic Symmetric Filter Mask", 0};
  Enumeration_knob(f, &_stageOfAlgorithm, stageLabels, "Output stage of algorithm:
     ");
1243  Tooltip(f, "The stage of the algorithm that you want to output.");
  SetFlags(f, Knob::STARTLINE);
1245
  Divider(f);
1247
  Int_knob(f, &_startFrameNumber, "Start Frame: ");
1249  Tooltip(f, "The Start Frame for video segmentation.");
  SetFlags(f, Knob::EARLY_STORE);
```

```
1251
    Int_knob(f, &_endFrameNumber, "End Frame: ");
1253 Tooltip(f, "The End Frame for video segmentation.");
    SetFlags(f, Knob::EARLY_STORE);
1255
    Divider(f);
1257
    Bool_knob(f, &_smoothProbHistogram, "Smooth the probability histogram");
1259 Tooltip(f, "Add smoothing to the probability histograms.");
    SetFlags(f, Knob::STARTLINE);
1261
    Bool_knob(f, &_blurProbMap, "Blur probability map");
1263 Tooltip(f, "Add a blur to the probability map.");
    SetFlags(f, Knob::STARTLINE);
1265
    Int_knob(f, &_blurKernelSize, "Kernel size: ");
1267 Tooltip(f, "The size of the kernel for the blur operation (1 is 3x3, 2 is 5x5
      etc.).");

1269 Bool_knob(f, &_fallOffProbMap, "Use Fall-off Probability Map: ");
    Tooltip(f, "Use probability map that 'falls-off' the further the pixel is from
      user strokes.");
1271 SetFlags(f, Knob::STARTLINE);

1273 Bool_knob(f, &_distanceSquared, "Distance squared in the prob map fall-off");
    Tooltip(f, "Use the the distance squared in the probability map fall-off.");
1275


1277 Divider(f);

1279 Float_knob(f, &_imageGradientStrength, "Image Gradient Strength: ");
    Tooltip(f, "The amount which the image gradient is taken into account in the
      distance transform.");
1281 SetRange(f, 0, 100);

1283 Bool_knob(f, &_useProbMapPenalty, "Use the probability map penalty");
    Tooltip(f, "The penalty for the geodesic distance is set by the probability map
      rather than the greyscale of the original image.");
1285 SetFlags(f, Knob::STARTLINE);

1287 Float_knob(f, &_probabilityMapStrength, "Probability Map Strength: ");
    Tooltip(f, "The amount which the probability map constructed from the foreground
       and background marks is taken into account in the distance transform.");
1289 SetRange(f, 0, 100);

1291 const char* probMapLabels[] = { "Colour", "Motion", "Mixed", "Texture", 0 };
    Enumeration_knob(f, &_probMapType, probMapLabels, "Probability Map Type: ");
1293 Tooltip(f, "Select which probability map to use, either based on colour or
      motion.");
    SetFlags(f, Knob::STARTLINE);
1295
```

```
    Bool_knob(f, &_useMinProbValue, "Use Minimum Prob Value: ");
1297 Tooltip(f, "Use the minimum probability value between the colour or motion maps.
       ");

1299 Float_knob(f, &_colourMotionMix, "Strength of colour to motion probability maps
       (0-1): ");
    Tooltip(f, "The strength of the colour probability map to the motion probability
       map (0-1).");
1301 SetRange(f, 0, 1);

1303 Bool_knob(f, &_initialiseWithProbMap, "Initialise with Probability Map");
    Tooltip(f, "Initialise the distance transform with the probability map.");
1305 SetFlags(f, Knob::STARTLINE);

1307 Float_knob(f, &_sigmoidTransformationFactor, "Sigmoid Tranformation Factor:");
    Tooltip(f, "A variable of the sigmoid transformation used in the probability map
       .");
1309 SetRange(f, 1, 10);

1311 Int_knob(f, &_noRasterScans, "No. of Raster Scans for Distance Transform: ");
    Tooltip(f, "The number of times the distance transforms will scan the image to
       calculate the distance values from the object area");
1313
    Bool_knob(f, &_runSuperScan, "Run Super Scan");
1315 Tooltip(f, "Run the more detailed scan over the video object");

1317 Int_knob(f, &_erosionPixels, "Erosion Pixels: ");
    Tooltip(f, "Number of pixels used in the geodesic erosion calculation.");
1319
    Bool_knob(f, &_swapErosionTrueFalse, "Swap erosion true false");
1321 Tooltip(f, "Swap erosion true false.");

1323 Int_knob(f, &_dilationPixels, "Dilation Pixels: ");
    Tooltip(f, "Number of pixels used in the geodesic dilation calculation.");
1325
    Divider(f);
1327
    Float_knob(f, &_opticalFlowStrength, "Optical Flow Divergence Strength: ");
1329 Tooltip(f, "The amount which the optical flow divergence is taken into account
       in the distance transform.");
    SetRange(f, 0, 100);
1331
    const char* _divergenceCalculationLabels[] = { "1 block compare", "2 block
       compare", "Kernel compare", 0 };
1333 Enumeration_knob(f, &_divergenceMethod, _divergenceCalculationLabels, "
       Divergence calculation method");
    Tooltip(f, "The comparison method used to calculate the divergence.");
1335
    Int_knob(f, &_kernelSizeInput, "Kernel size radius");
1337 Tooltip(f, "The radius size of kernel used in comparison: 1 is a 3x3 kernel, 2
       is a 5x5 kernel etc.");
```

```
1339    Divider(f);
1341    Bool_knob(f, &_autoMoveUserInputFG, "Auto adjust foreground user input using
          motion vectors");
        Tooltip(f, "Auto-move the foreground user input across frames using the values
          in the motion vectors generated.");
1343    SetFlags(f, Knob::STARTLINE);
1345    Bool_knob(f, &_autoMoveUserInputBG, "Auto adjust background user input using
          motion vectors");
        Tooltip(f, "Auto-move the background user input across frames using the values
          in the motion vectors generated.");
1347    SetFlags(f, Knob::STARTLINE);
1349
        Tab_knob(f, "Texture analysis");
1351
}
1353
/*
1355 * Make sure changing input updates the knobs.
 */
1357 int GeodesicVideoSegment::knob_changed(Knob* k)
{
1359   _firstTime = true;
    return 1; //Iop::knob_changed(k);
1361 }
1363 /*
 * this function creates an instance of GrabCut and is called from Iop::
     Description (below) when a node is created.
1365 */
static Iop* CreateGeodesicVideoSegmentNode(Node* node)
1367 {
    return new GeodesicVideoSegment(node);
1369 }
1371 /*
 * Iop::Description is how NUKE knows the name of the operator.
1373 */
const Iop::Description GeodesicVideoSegment::d(CLASS, "Segmentation/
    GeodesicVideoSegment", CreateGeodesicVideoSegmentNode);
1375
1377
// THREADING FUNCTIONS
1379
/*
1381 * Method added for threading the generalized geodesic distance transforms.
 */
```

```
1383 void GeodesicVideoSegment::ggdtsForSigned(unsigned index, unsigned nThreads, void*
       d)
     {
1385   if (index == 0)
       {
1387     //if it's the first thread calculate the ggdt
       ((GeodesicVideoSegment*)d)->_ggdt = ((GeodesicVideoSegment*)d)->
       _geodesicSegment->generalisedGeodesicDistanceTransform(true, true);
1389   }
       else if (index == 1)
1391   {
       //if it's the second thread calculate the inverse ggdt
1393   ((GeodesicVideoSegment*)d)->_inverseGGDT = ((GeodesicVideoSegment*)d)->
       _geodesicSegment->generalisedGeodesicDistanceTransform(false, true);
       }
1395   else
       {
1397   std::cout << "*** Threading error ***  For some reason there are more than 2
       threads...\n";
       }
1399 }


1401
     /*
1403  * Method added for threading the erosion and dilation geodesic distance
       transforms.
      */
1405 void GeodesicVideoSegment::ggdtsForSymmetricSigned(unsigned index, unsigned
       nThreads, void* d)
     {
1407   int _frameHeight = ((GeodesicVideoSegment*)d)->_imageHeight;
       int _frameWidth= ((GeodesicVideoSegment*)d)->_imageWidth;
1409   int _numberOfFrames = ((GeodesicVideoSegment*)d)->_numberOfFrames;

1411   if (index == 0)
       {
1413   //erosion
       //filter signed ggdt to create a new mask
1415   ((GeodesicVideoSegment*)d)->_erosionMask = new Array3D<bool>(_frameWidth,
       _frameHeight, _numberOfFrames);
       for (int frameNumber = 0; frameNumber < _numberOfFrames; frameNumber++)
1417   {
         for (int row = 0; row < _frameHeight; row++)
1419     {
           for (int col = 0; col < _frameWidth; col++)
1421       {
             //update the mask depending on the value in the signed GGDT
1423         if (((GeodesicVideoSegment*)d)->_signedGGDT->getArrayValue(row, col,
       frameNumber) > -(((GeodesicVideoSegment*)d)->_erosionPixels))
             {
1425           ((GeodesicVideoSegment*)d)->_erosionMask->setArrayValue(row, col,
```

```
    frameNumber, !((GeodesicVideoSegment*)d)->_swapErosionTrueFalse ? true : false
    );//true
          }
1427      else
          {
1429        ((GeodesicVideoSegment*)d)->_erosionMask->setArrayValue(row, col,
    frameNumber, !((GeodesicVideoSegment*)d)->_swapErosionTrueFalse ? false : true
    );//false
          }
1431    }
      }
1433  }
    ((GeodesicVideoSegment*)d)->_erosionGGDT = ((GeodesicVideoSegment*)d)->
    _geodesicSegment->generalisedGeodesicDistanceTransform(true, ((
    GeodesicVideoSegment*)d)->_erosionMask, true, true);
1435 }
  else if (index == 1)
1437 {
    //dilation
1439  //create new mask object
    ((GeodesicVideoSegment*)d)->_dilationMask = new Array3D<bool>(_frameWidth,
    _frameHeight, _numberOfFrames);
1441  for (int frameNumber = 0; frameNumber < _numberOfFrames; frameNumber++)
    {
1443    for (int row = 0; row < _frameHeight; row++)
      {
1445      for (int col = 0; col < _frameWidth; col++)
        {
1447      if (((GeodesicVideoSegment*)d)->_signedGGDT->getArrayValue(row, col,
    frameNumber) > ((GeodesicVideoSegment*)d)->_dilationPixels)
          {
1449        ((GeodesicVideoSegment*)d)->_dilationMask->setArrayValue(row, col,
    frameNumber, !((GeodesicVideoSegment*)d)->_swapErosionTrueFalse ? true : false
    );
          }
1451      else
          {
1453        ((GeodesicVideoSegment*)d)->_dilationMask->setArrayValue(row, col,
    frameNumber, !((GeodesicVideoSegment*)d)->_swapErosionTrueFalse ? false : true
    );
          }
1455    }
      }
1457  }
    //set useObjectMap to false as this needs to be the complement of the dilation
     mask.
1459  ((GeodesicVideoSegment*)d)->_dilationGGDT = ((GeodesicVideoSegment*)d)->
    _geodesicSegment->generalisedGeodesicDistanceTransform(false, ((
    GeodesicVideoSegment*)d)->_dilationMask, true, true);
  }
1461 else
```

```
  {
1463    std::cout << "*** Threading error ***  For some reason there are more than 2
        threads...\n";
  }
1465 }
```

Listing B.1: Source code for the main Nuke plugin class which uses colour and motion information to segment objects over video frames. Further classes such as GeodesicSegment, Image and Video provide the functionality required by the plugin.